NCT® 3xxT

Control for Lathes

Programmer's Manual

From SW version n.16.0

Manufactured by **NCT Automation kft**.
H-1148 Budapest Fogarasi út 7
⊠ Mailing address: H-1631 Bp. pf.26
☎ Phone: (+36 1) 467 6300
☎ Fax:(+36 1) 467 6309
E-mail: nct@nct.hu
Home page: www.nct.hu

ATTENTION!

☞*Note:* Please read the changes carefully before using new software!

New features and differences of software version 16 compared to version 15

10.2.3 Compensating the Angular Position of the Workpiece: on page 87
The misalignment compensation can now be specified along all three rotary axes at the same time, and is not limited to just one axis.

18.2.1 Roughing Cycle (G71): on page 197
The type 3 roughing has been introduced in the roughing cycle (see on page 206).

18.2.2 Face Roughing Cycle (G72): on page 208
The type 3 face roughing has been introduced in the face roughing cycle (see on page 211).

20.7 Managing Non-perpendicular Axes: on page 271
If the movement of one axis on a machine is not perpendicular to the other one, but makes an angle with respect to the right angle, this function should be used.

15.1 Compensation Memory. Referring to Tool Compensation (T or D): on page 123
The separate lathe and mill compensation memory in the controller has been eliminated. There is now one common memory. If one channel is set to lathe and the other to mill, they have a common compensation memory.

23.1.16 Value of the Actual Length Compensation: on page 309

23.1.18 Values of the Tool Compensation Memory: on page 310

Contents

1 Introduction. . . 11
1.1 Part Program. . . 11
1.2 Channel. . . 12
1.3 Fundamental Terms. . . 13

2 Controlled Axes. . . 17
2.1 Naming and Numbering the Axes. . . 17
2.2 Extended Axis Names. . . 18
2.3 Assigning Axes to Channels. . . 18
2.4 Unit System of Axes and Accuracy of Position Display. . . 19

3 Preparatory Functions (Codes G). . . 20

4 Interpolation. . . 25
4.1 Positioning (G0). . . 25
4.1.1 Positioning by Linear Interpolation. . . 25
4.1.2 Positioning by Overlapping the Rapid Traverse Motions. . . 26
4.2 Linear Interpolation (G1). . . 27
4.3 Circular Interpolation (G2, G3). . . 29
4.3.1 Planar Spiral Interpolation (G2, G3). . . 34
4.3.2 Helical Spiral Interpolation (G2, G3). . . 36
4.3.3 Conical Spiral Interpolation (G2, G3). . . 39
4.4 Equal Lead Thread Cutting (G33). . . 41
4.5 Polar Coordinate Interpolation (G12.1, G13.1). . . 43
4.6 Cylindrical Interpolation (G7.1). . . 48

5 Coordinate Data. . . 51
5.1 Absolute and Incremental Programming (G90, G91). . . 51
5.2 Inch/Metric Conversion (G20, G21). . . 52
5.3 Programming in Diameter or Radius. . . 52
5.3.1 Switching between in Radius and in Diameter Programming (G10.9). . . 53
5.4 Data Specification in Polar Coordinates (G15, G16). . . 54
5.5 Specification and Accuracy of Coordinate Data. . . 56
5.6 Managing the Rotary Axis Roll-Over. . . 57

6 Feed. . . 63
6.1 Rapid Traverse. . . 63
6.2 Cutting Feed. . . 63
6.2.1 Feed per Minute (G94) and Feed per Revolution (G95). . . 64
6.3 Feed Control Functions. . . 64
6.3.1 Exact Stop at the End of the Block (G9). . . 65
6.3.2 Exact Stop Mode (G61). . . 65
6.3.3 Continuous Cutting Mode (G64). . . 65
6.3.4 Override and Stop Inhibition Mode (G63). . . 65
6.3.5 Automatic Feed Override at Inner Corners (G62). . . 65
6.4 Automatic Feed Override in the Cases of Inner Circle Arcs. . . 66

7 Acceleration 68
7.1 Automatic Deceleration at Corners in the State G64. 71
7.2 Limiting the Normal Direction Accelerations. 74
7.3 Limiting the Acceleration Step Change (Jerk). 76

8 Dwell (G4). 78

9 Reference Point. 79
9.1 Automatic Reference Point Return (G28). 80
9.2 Return to the 2nd, 3rd and 4th Reference Point (G30). 81

10 Coordinate Systems and Plane Selection. 83
10.1 Machine Coordinate System. 83
10.1.1 Positioning in the Machine Coordinate System (G53). 84
10.2 Workpiece Coordinate Systems. 84
10.2.1 Selecting the Workpiece Coordinate System (G54...G59). 85
10.2.2 Selecting the Additional Workpiece Coordinate Systems (G54.1 P). 87
10.2.3 Compensating the Angular Position of the Workpiece. 87
10.2.4 Setting the Offset of the Workpiece Coordinate Systems (G10 L2). 89
10.2.5 Setting the Offset of the Additional Workpiece Coordinate Systems (G10 L20) 90
10.2.6 Creating a New Work Coordinate System (G92). 90
10.3 Local Coordinate System (G52). 91
10.4 Plane Selection (G17, G18, G19). 94

11 Spindle Functions. 96
11.1 Spindle Speed Command (Code S). 96
11.1.1 Referring to Several Spindles. Extending the Address S. 96
11.1.2 Assigning Spindles to Channels. 97
11.2 Functions M Controlling the Spindle. 97
11.3 Managing the Speed Ranges. 98
11.4 Main Spindle. Selecting the Main Spindle. 99
11.5 Controlling the Constant Surface Speed. 100
11.5.1 Specifying the Constant Surface Speed Control (G96 S, G97 S). 100
11.5.2 Clamping the Speed during Calculation of Constant Surface Speed (G92 S) 101
11.5.3 Selecting an Axis for Constant Surface Speed Control (G96 P). 101
11.6 Spindle Speed Fluctuation Detection. 102
11.7 Positioning the Spindles. 102
11.7.1 Spindle Orientation. 102
11.7.2 Stopping the Spindles and Closing the Position Control Loop. 103
11.7.3 Programming of the Positioning the Spindles. 103
11.7.4 Position-correct Synchronization of Two Spindles. 105
11.7.5 Turning the Position-controlled Operating Mode off. 106
11.8 Converting Spindle into Axis and Axis into Spindle. 107

12 Function T. 109
12.1 Programming the Tool Change. 109

13 Miscellaneous and Auxiliary Functions. . . . 111
13.1 Miscellaneous Functions (Codes M). . . . 111
13.2 Auxiliary Functions (A, B, C, U, V or W). . . . 113
13.3 Buffer Emptying Functions. . . . 114

14 Part Program Configuration. . . . 115
14.1 Block Number (Address N). . . . 115
14.2 Conditional Block Skip (/ address). . . . 115
14.3 Writing Comments into the Part Program: (comment). . . . 116
14.4 Main Program and Subprogram. . . . 116
14.4.1 Identification of Programs in Memory. The Program Number (O). . . . 117
14.4.2 Calling a Subprogram (M98). . . . 117
14.4.3 Return from a Subprogram (M99). . . . 119
14.4.4 Jump within the Main Program. . . . 121
14.5 Functions M of Channel Synchronization. . . . 121

15 Tool Compensation. . . . 123
15.1 The Compensation Memory. Referring to Tool Compensation (T or D). . . . 123
15.2 The Second Geometry Compensation Memory. . . . 126
15.3 Modifying the Tool Compensation Values from the Program (G10). . . . 128
15.4 Tool Length Compensation by Code T. . . . 128
15.5 Tool Length Compensation by Code G (G43.7, G49). . . . 131
15.6 Tool Nose Radius Compensation (G40, G41, G42). . . . 133
15.6.1 Start up of the Tool Nose Radius Compensation. Moving to the Contour. . . . 136
15.6.2 Calculation of Tool Nose Radius Compensation in Offset Mode. . . . 143
15.6.3 Cancelling the Tool Nose Radius Compensation. Leaving the Contour. . . . 150
15.6.4 Reversal in Calculation of Tool Nose Radius Compensation. . . . 155
15.6.5 Programming Vector Preservation (G38). . . . 158
15.6.6 Programming Corner Arc (G39). . . . 159
15.6.7 Troubles in Tracking the Contour. Interference Check. . . . 161

16 Special Transformations. . . . 169
16.1 Rotating a Shape Around a Given Point (G68, G69). . . . 169
16.2 Scaling a Shape in Relation to a Given Point (G50, G51). . . . 171
16.3 Mirroring a Shape through One or More Straight Lines (G50.1, G51.1). . . . 174
16.4 Programming Rules for Specific Transformations. . . . 176

17 Automatic Geometric Calculations. . . . 178
17.1 Programming Chamfer and Corner Rounding. . . . 178
17.2 Specification of a Straight Line Using its Angle of Inclination. . . . 179
17.3 Calculations of Intersection Point in the Plane. . . . 181
17.3.1 Linear-Linear Intersection. . . . 181
17.3.2 Linear-Circular Intersection. . . . 183
17.3.3 Circular-Linear Intersection. . . . 185
17.3.4 Circular-Circular Intersection. . . . 187
17.3.5 Chaining the Intersection Calculations. . . . 189

18 Canned Cycles for Turning. . . . 190

18.1 Single Cycles. 190
18.1.1 Longitudinal Turning Cycle (G77).. 190
18.1.2 Simple Thread Turning Cycle (G78).. 192
18.1.3 Face Turning Cycle (G79). 194
18.1.4 Application of Single Cycles. 196
18.2 Multiple Repetitive Cycles. 197
18.2.1 Roughing Cycle (G71). 197
18.2.2 Face Roughing Cycle (G72).. 208
18.2.3 Pattern Repeating Cycle (G73).. 214
18.2.4 Finishing Cycle (G70). 217
18.2.5 Face Grooving Cycle (G74). 218
18.2.6 Grooving Cycle (G75). 220
18.2.7 Multiple Threading Cycle (G76). 222

19 Canned Cycles for Drilling. 228
19.1 Detailed Description of the Drilling Cycles. 235
19.1.1 High-speed Peck Drilling Cycle (G83.1). 235
19.1.2 Left-Handed Tapping Cycle Using Spring Tap (G84.1). 236
19.1.3 Boring Cycle with Automatic Tool Shift (G86.1). 237
19.1.4 Cancelling the Cycle State (G80).. 238
19.1.5 Drilling Cycle with Retraction at Rapid Traverse Rate (G81).. 238
19.1.6 Drilling Cycle with Dwell and with Retraction at Rapid Traverse (G82). . . . 239
19.1.7 Peck Drilling Cycle (G83). 240
19.1.8 Right-Handed Tapping Cycle Using Spring Tap (G84).. 241
19.1.9 Rigid Tapping Cycle (G84.2, G84.3).. 242
19.1.10 Peck Rigid Tapping Cycle (G84.2, G84.3). 244
19.1.11 Boring Cycle with Retraction at Feed Rate (G85). 247
19.1.12 Boring Cycle with Retraction with Spindle in Standstill (G86).. 248
19.1.13 Manual Control/Back Boring Cycle (G87). 249
19.1.14 Boring Cycle with Dwell and Manual Operation at the Bottom (G88). 251
19.1.15 Boring Cycle with Dwell and with Retraction at Feed Rate (G89). 252
19.2 Remarks on the Use of the Drilling Cycles.. 253

20 Functions to Control Axes. 254
20.1 Polygonal Turning.. 254
20.1.1 Principle of Poligonal Turning.. 254
20.1.2 Programming the Polygonal Turning (G51.2, G50.2). 256
20.2 Gear Hobbing (G81.8). 259
20.3 Synchronous Control of Axes.. 259
20.4 Interchanging of Axes.. 262
20.5 Superimposed Control of Axes.. 267
20.6 Changing the Axis Direction. 269
20.7 Managing Non-perpendicular Axes. 271

21 Measurement Functions. 277
21.1 Skip Function (G31). 277
21.2 Torque Limit Skip (G31). 278
21.3 Automatic Tool Length Measurement (G36, G37).. 280

22 Safety Functions. 282
22.1 Stroke End. 282
22.2 Working Area Limitation from Parameter/Program (G22, G23). 283
22.3 The Area Forbidden Internally. 286
22.4 Monitoring the Forbidden Area Prior to Motion Start. 286

23 Custom Macro. 289
23.1 Variables of the Programming Language. 290
23.1.1 Referring to Variables. 290
23.1.2 Number Representation of Macro Variables. 291
23.1.3 Local Variables: #1 – #33. 291
23.1.4 Common Variables. 292
23.1.5 Notation Used in Description of System Variables. 292
23.1.6 Vacant Variable. Constants. 293
23.1.7 Variables Between the Part Program and the PLC Program. 294
23.1.8 Messages of the Part Program. 295
23.1.9 Clock, Timers and Part Counters.. 297
23.1.10 Variables Influencing the Operation of the Automatic Mode. 299
23.1.11 Querying the Block Search and Test Statuses. 300
23.1.12 Status of Mirror Image.. 301
23.1.13 Number of the Main Program. 301
23.1.14 Modal Information.. 302
23.1.15 Position Information. 307
23.1.16 Value of the Actual Length Compensation. 309
23.1.17 Other Position Information. 309
23.1.18 Values of the Tool Compensation Memory.. 310
23.1.19 Workpiece Zero Point Offsets.. 311
23.1.20 Reading Data of Tools Being in Spindle and in Stand-by Magazines.. 315
23.1.21 Reading the Data of the Pallet Being in the Working Space and in the Loading-Unloading Point. 317
23.2 Instructions of the Program Language. 318
23.2.1 Definition or Replacement.. 318
23.2.2 Arithmetic Operations. 318
23.2.3 Logic Operations. 319
23.2.4 Functions. 321
23.2.5 Conversion Instruction.. 322
23.2.6 Execution Sequence of Complex Arithmetic Operations. 323
23.2.7 Conditional Expressions. 324
23.2.8 Unconditional Branch: 324
23.2.9 Conditional Branch. 324
23.2.10 Conditional Instruction. 325
23.2.11 Iteration. 325
23.2.12 Indirect Axis Address Specification. 329
23.2.13 Data Output Commands. 329
23.3 Calling the Macros, System Macros and System Subprograms.. 336
23.3.1 Simple Macro Call (G65). 339
23.3.2 Macro Modal Call after Each Motion Command (G66). 341
23.3.3 Macro Modal Call from Each Block (G66.1). 342
23.3.4 System Macro Call by Codes G Given in Parameter.. 344

23.3.5 System Macro Call by Codes M Given in Parameter. 346
23.3.6 System Subprogram Call by the Code M Given in Parameter. 348
23.3.7 System Subprogram Call by Codes A, B, C, S, T Enabled in Parameter. 350
23.3.8 System Subprogram Call by Codes ASCII Given in Parameter. 351
23.3.9 Displaying the Blocks of Macros and Subprograms in Automatic Mode. 352
23.4 Interruption-type Macro. 353
23.5 NC and Macro Instructions. Execution of Macro Blocks. 357
23.6 Pocket Milling Macro Cycle. 359

24 Writing and Reading the Parameters. 363
24.1 Writing the Parameters from Part Program (G10 L52). 364
24.2 Reading the Parameters from Part Program (PRM). 365

Notes. 367

January 31, 2024

1 Introduction

1.1 Part Program

The Part Program is a set of instructions that can be interpreted by the control system in order to control the operation of the machine.
The Part Program consists of blocks which, in turn, comprise words.

Word: Address and Data
Each word is made up of two parts - an address and a data. The address has one or more characters, the data is a numerical value (an integer or decimal value). Some addresses may be given a sign or operator I. In the program, addresses can be given using small letter or capital letter too, the control accepts both variants.

Address Chain:

Address	Meaning	Value limits
O	program number	0001 - 9999
/	optional block	1 - 9
N	block number	1 - 99999999
G	preparatory function	*
X, Y, Z, U, V ,W	length coordinates	I, −, *
A, B, C,	angular coordinates	I, −, *
R	circle radius, auxiliary data	I, −, *
I, J, K	circle center coordinate, auxiliary coordinate	−, *
E	auxiliary coordinate	−, *
F	feed rate	*
S	spindle speed	*
M	miscellaneous function	1 - 99999999
T	tool number/compensation number	1 - 99999999
D	compensation number	1 - 999
L	repetition number	1 - 99999999
P	auxiliary data, dwell time	0
Q	auxiliary data	0
,C	distance of chamfer	−, *
,R	radius of fillet	−, *
,A	angle of straight line	−, *

At an address marked with a * in the Value Limits column, the data may have a decimal value as well.
At an address marked with **I** and –, an incremental operator or a sign can be assigned, respectiv-

ely.
The positive sign + is not indicated and not stored.

Block

A block is made up of words.
The blocks are separated by characters ␊ (**L**ine **F**eed) in the memory. The use of a block number is not mandatory in the blocks.

Main Program and Subprogram

The part programs may be divided into two main groups:
main programs, and
subprograms.
The procedure of machining a part is described in the main program. If, in the course of machining repeated patterns should be machined at different places, it will not be necessary to write those program-sections over and over again in the main program, instead, a subprogram will have to be organized, which can be called from any place (even from another subprogram). The user can return from the subprogram to the calling program.

Program Format in the Memory

The ***part programs*** stored in the memory ***are ASCII code text files***.

1.2 Channel

In general, at one time on one machine one tool is working motion of which is controlled by the part program through the control.
We speak about multi-channel operation, when the same control simultaneously directs the path of two or more tools independent of the other's motion, executing two or more part .
For each channel it can be assigned a part program to execute.
Multi-channel operation is required too, when at a time only one tool is working, but on the machine there is a workpiece feeder for which motions of the workpiece should be programmed while the tool is cutting.
Synchronization points can be programmed for those points of the programs, where ***tool path motions should wait for each other***.
Each channel has its own workpiece zero point table, tool compensation table and macro variables. The part of tool compensation table and the macro variables assignable on one parameter can be made common for each channel.
In standard version, the NCT2xx controls are single-channel ones, however, optionally, they are able to function as multi-channel ones. As a maximum, 8 channels can be built in a control.
In the control, for each channel, the mode of channel operation can be specified in parameter. In case of the NCT200 controls there could be ***two ones: lathe channel or milling machine channel***. Within a control, the channels can be mixed. For example:
Channel 1: lathe
Channel 2: lathe
Channel 3: milling machine
Hereafter, this manual describes programming the lathe channel.
If milling machine channel is also built in the given control, description of programming for it can be found in the NCT2xxM Control for Milling Machines and Machining Centers Programmer's Manual.

1.3 Fundamental Terms

Interpolation

The control system can move the tool along straight lines and arcs in the course of machining. These activities will be hereafter referred to as "interpolation".

Tool movement along a straight line:

program:

```
        G01 X__ Z__
```

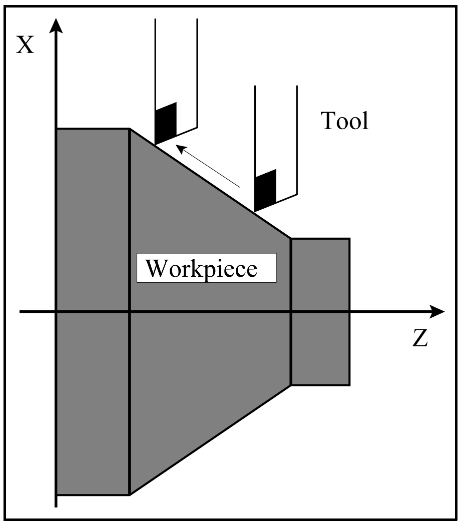


Fig. 1.3-1

Tool movement along an arc:

program:

```
        G02 X__ Z__ R__
```

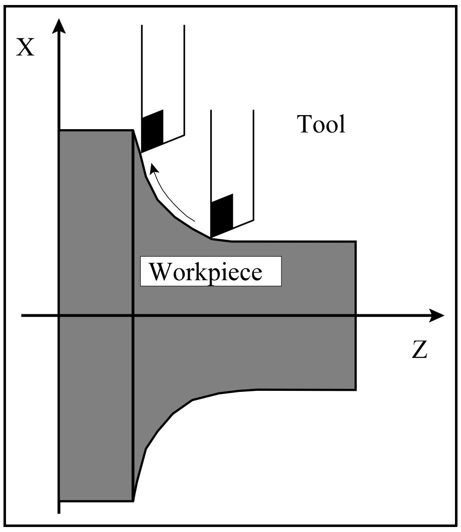


Fig. 1.3-2

Preparatory Functions (codes G)

The type of activity to be performed by a given block is described with the use of preparatory functions (also referred to as codes G). E.g., code G01 introduces a linear interpolation.

Feed

The term "feed" refers to the speed of the tool relative to the workpiece during the process of cutting. The desired feed can be specified in the program at address F and with a numerical value. For example F2 means 2 mm/rev.

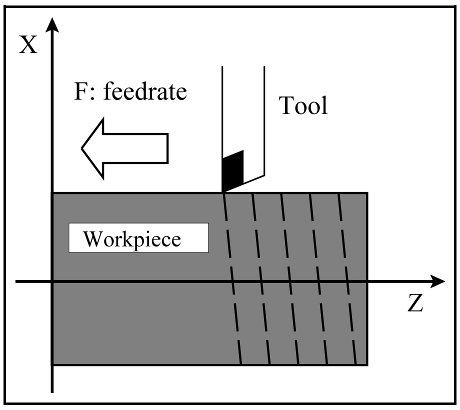


Fig. 1.3-3

Reference Point

The reference point is a fixed point on the machine tool. After power-on of the machine, the slides have to be moved to the reference point. Afterwards the control system will be able to interpret data of absolute coordinates as well.

Coordinate System

The dimensions indicated in the part drawing are measured from a given point of the part. That point is the origin of the workpiece coordinate system. Those dimensional data must be written at the coordinate address in the part program. For example, X150 Z–100 means a coordinate point of 150 and –100 mm in the coordinate system of the workpiece in the direction X and Z, respectively.

In order that the control can interpret programmed coordinate data, the distance between the machine zero point and the workpiece zero point must be given. This has to be done by measuring the workpiece zero point.

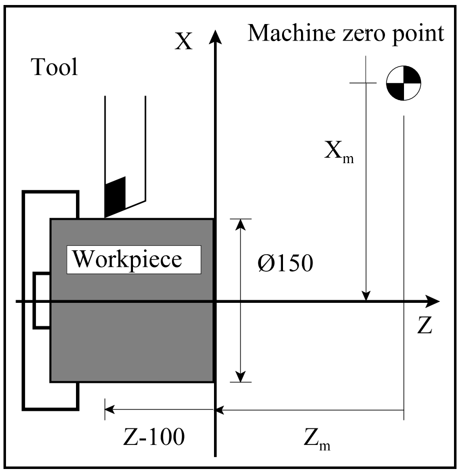


Fig. 1.3-4

Absolute Coordinate Specification

When absolute coordinates are specified, the tool travels a distance measured from the origin of the coordinate system, i.e. to a point whose position has been specified by the coordinates.

The code of absolute data specification is G90.

The instruction line

```
G90 X150 Z2
```

will move the tool to a point of the above position, irrespective of its position before the command has been issued .

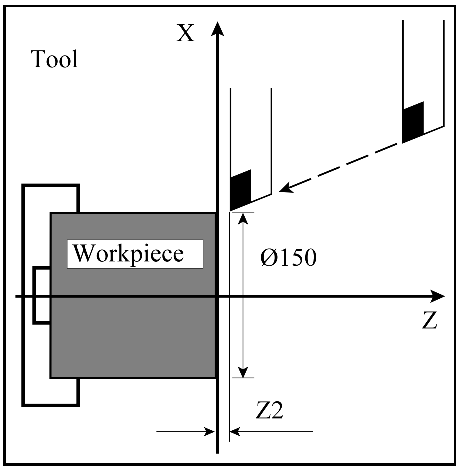


Fig. 1.3-5

Incremental Coordinate Specification

In the case of an incremental data specification, the control system will interpret the coordinate data in such a way that the tool will travel a distance measured from its instantaneous position.
The code of incremental data specification is G91. Code G91 refers to all the coordinate values.
The instruction line
```
G91 X-50 Z-125
```
will move the tool over the above distance from its preceding position.

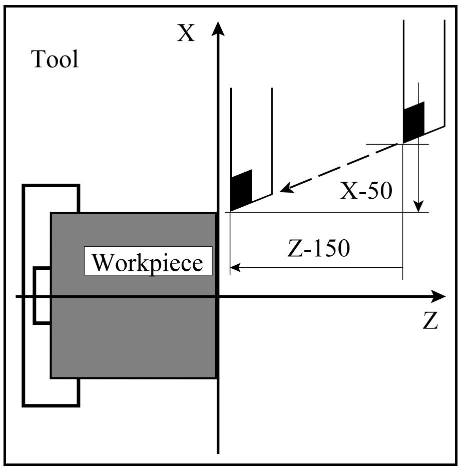


Fig. 1.3-6

Diameter Programming

The dimension of the X axis can be programmed in diameter too, on the basis of setting parameter.

Modal Functions

In the programming language, some instructions stays in effect or their value is valid until a command of contrary meaning is issued or different value is given to the adequate function. For example: In the program detail
```
N15 G90 G1 X20 Z30 F0.2
N16 X30
N17 Z100
```
the state of G90 (absolute data specification) and G1 (linear interpolation) and the value of F specified in the block N15 will be modal ones in the blocks N16 and N17. Thus, it is not necessary to specify these codes block by block.

Non-modal (One-shot) Functions

The effect of some functions or the values of data are valid in a given block. These functions are non-modal or one-shot ones.

Spindle Speed Command

The spindle speed can be specified at address S. It is also named S function. Instruction S1500 tells the spindle to rotate at a speed of 1500 rpm.

Constant Surface Speed Control

The control changes the spindle speed automatically depending on the diameter so that the speed of the tool tip relative to the surface of the workpiece will be constant. This function is the constant surface speed control.

Tool Function

In the course of machining, different tools have to be used for various cutting operations. Numbers are used to differentiate tools from each other. Tools can be referred by code T. The first two digits of the code T means the code of the tool (i.e. the number of position in which it can be found), and the last two digits represent the number of the compensation group belonging to the selected tool. The instruction

```
        T0212
```

in the program means that the tool No.2 has been selected and the compensation group No.12 has been assigned to it.

Miscellaneous Functions

A number of switching operations, e.g. starting the spindle, turning on the coolant have to be carried out in the course of machining. These operations can be executed by M (miscellaneous) functions. For example: In the series of instructions

```
        M3 M8
```

M3 means rotation of the spindle clockwise, M8 means turning on the coolant.

Tool Length Compensation

In the course of machining, tools of different length are used for the various operations. On the other hand, in production of a bigger series the same operation also has to be performed with tools of different lengths (e.g. when the tool breaks). In order to make the motions described in the part program independent of the length of the tool, i.e. of its offset, the various tool lengths must be set in the control system. For this, the length of the tools have to be measured. If the program is intended to move the tip of the tool to the specified point, the value of the particular length data gained and given by measurement will have to be called. This is feasible at the second two digits of the code T . Henceforth the control will move the tip of the tool to the specified point.

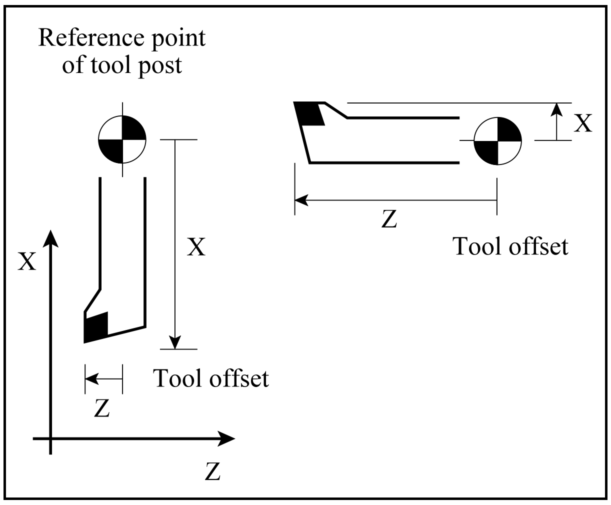


Fig. 1.3-7

Tool Nose Radius Compensation

In the case of turning a contour, and when the motion of the tool is not parallel with the axes, exact size can be achieved guiding not the tip of the tool along the contour, but the center of the tool nose along the path perpendicular distance of which is equal to the radius (r). In order to write in the program not the path of the tool nose center taking the tool radius into account, but the real contour data of the workpiece, introduction of radius compensation is necessary. In the program, the tool radius must be set in the compensation group called at the address T.

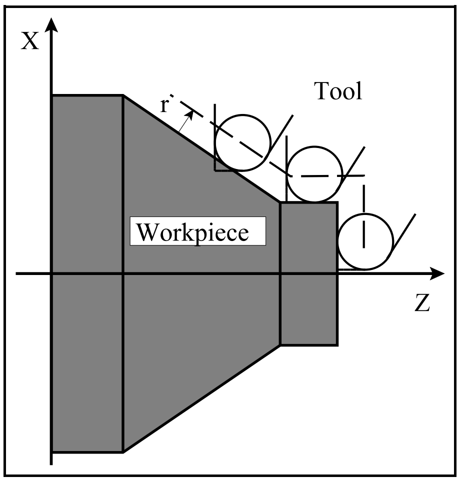


Fig. 1.3-8

2 Controlled Axes

Number of axes in basic configuration	2 axes
Number of expansion axes	Maximum 14 additional axes in the same channel
Maximum number of axes	32 axes altogether in several channels

2.1 Naming and Numbering the Axes

The ***names of the controlled axes*** can be defined in the parameter memory. It can be assigned here, which physical axis has to move to which address.

In the basic configuration, the names of axes in the lathe control are: X and Z.

These axes will be set in the parameter N0103 Axis to Plane as ***main axes***.

The names of the expansion axes depends on type of the given axis.

If on the lathe the axis Y is also built in as an expansion axis, the axis Y will have to be set as a main axis too in the parameter N0103 Axis to Plane.

Possible names of additional expansion axes performing linear motions are: U, V and W. If these axes are parallel with one of the main directions, the names of the ***expansion axes parallel*** with the axes X, Y and Z will be U, V and W, respectively.

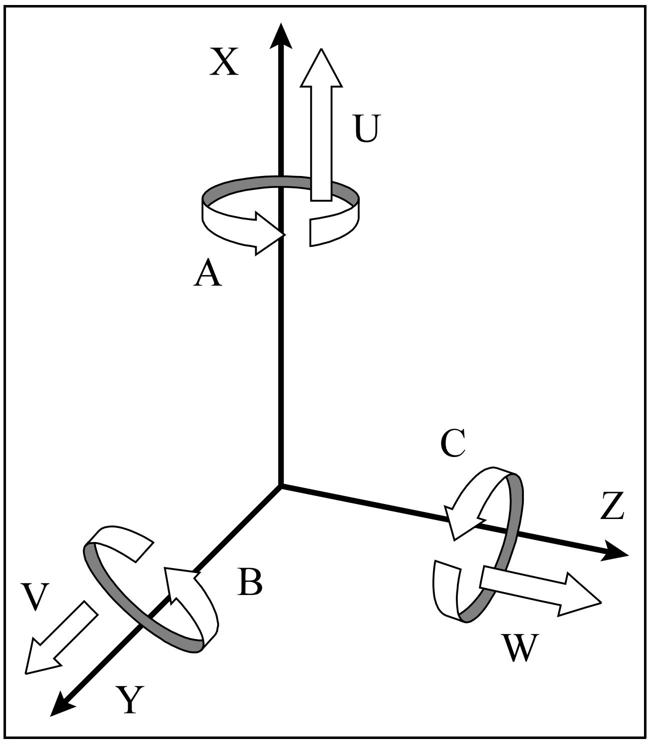


Fig. 2.1-1

Whether a linear expansion axis is parallel with a basic axis or not, it can be set in the parameter N0103 Axis to Plane.

The names of the axes performing rotational motions are: A, B and C. The names of the rotary axes parallel with the directions X, Y and Z will be A, B and C, respectively. Whether an axis is a rotary one or not, it can be set by ROT=1 in the parameter N0106 Axis Properties.

The axes are registered by the control **on the basis of their numbers**. The names of the axes must be assigned to the numbers of the axes. Usually, the axis X is set as the axis 1, the axis Y is set as the axis 2, the axis Z is set as the axis 3.

In certain program instructions, for example in case of referring to the macro variable querying the axis position, not the name of the axis, but the number of the axis must be given.

For numbering please ask the machine tool builder.

2.2 Extended Axis Names

In the case of complex multi-axis machines, the abovementioned 9 letters are not enough to name all the axes. For this reason, extended axis names have been introduced **to name an axis using** not one letter but **up to three characters**.
The first character must be X, Y, Z, U, V, W, A, B, or C necessarily, and it can be specified in the parameter N0100 Axis Name1. This parameter will have to be filled in even if one-character axis names are used only.
The second and the third character names can be specified in the parameter N0101 Axis Name2 and in the parameter N0102 Axis Name3, respectively. They could be the letters of the English alphabet: A, B, C, D, ... Y, Z, or numbers: 0, 1, 2, ..., 9. If the second or third axis name is not used, the value of the parameter will be 0.
Thus, the name of an axis could be XDE, but Z1 and Z2 could also be used.
If the name of the axis ends in a letter, its inherent value could be written next to it. The meaning of

```
        XDE127.81
```

is: the axis XDE has to go to the position 127.8.
If the name of the axis ends in a number, equality symbol = will have to be written after the axis. The meaning of

```
        Z1=87.257
```

is: the axis Z1 has to go to the position 87.257.
Certainly, even if using extended axis names, it will have to be specified in the parameter N0103 Axis to Plane which ones are main or basic axes, and which ones are parallel axes.
Hereafter in this manual, one-character axis names will be used, in general.

2.3 Assigning Axes to Channels

Using parameters, the builder of the machine tool assigns each axis to different channels. These parameter settings mean the state after the turning on.
The ***axes*** are always assigned to the channels ***in accordance with their number***. So, ***axis number*** in the control is ***global***, while ***axis names*** are ***local***, they are assigned to channels. Certainly, axes with the same name are not allowed within one channel, on the other hand they could be in different channels. The control is capable of managing maximum 16 axes being within one channel.
For example:

Channel 1:	Channel 2:
Axis1: X	Axis 5: X
Axis 2: Y	Axis 6: Y
Axis 3: Z	Axis 7: Z
Axis 4: C	Axis 8: C

In the course of the machining, it could be necessary to use one or more axes in another channel. In this case, it is allowed to interchange two axes between two channels, or to transfer one axis to another channel. The interchange is executed in accordance with the axis number. After change, the axis name may remain, or may change too.
The interchange of axes is realized by the builder of the machine tool through the PLC program, for example using M function. Description of them is always contained in the manual of the given machine tool.

2.4 Unit System of Axes and Accuracy of Position Display

Coordinate data can be given with an ***accuracy of 15 digit*** as maximum. The decimal point must be put in the only case, when positioning to a point of non-integer coordinate has to be performed. Coordinate data could also have sign. The sign + is omitted.
In the program, data of length coordinates can be given either in ***mm*** or in ***inch***. This is the ***input unit system***. The input unit system can be selected from the program, using code G (G21/G20).
The path-measuring device mounted on the machine tool can measure position either in mm or in inch. The path-measuring device determines the ***output unit system*** which must be given at the bit IND of the parameter N0104 Unit of Measure of the control. On a given machine tool, it is not possible to mix output unit systems among the axes.
If the input and output unit systems are different ones, conversion will be performed by the control automatically.
The rotational axes are always provided with ***degrees*** as units of measure. ***The rotary axes can be designated by ROT=1 in the parameter N0106 Axis Properties. Correct setting of this parameter is important because the control does not execute inch/mm conversion for the axes designated in this way.***
The ***position display accuracy*** (i.e. number of decimal places) can be set in the parameter N0105 Increment System.
The inner position representation of the system is independent of the value of the Increment System parameter. The accuracy is as follows:

In case of linear axes and measurement in mm: 10^{-6} mm;
In case of linear axes and measurement in inch: 10^{-7} inch;
In case of circular axes: 10^{-6} degree.

System name	Axis	Display accuracy in metric and inch systems	
		G21 metric	**G20 inch**
ISA	linear	0.01 mm	0.001 in
	rotary	0.01 deg	0.01 deg
ISB	linear	0.001 mm	0.0001 in
	rotary	0.001 deg	0.001 deg
ISC	linear	0.0001 mm	0.00001 in
	rotary	0.0001 deg	0.0001 deg
ISD	linear	0.00001 mm	0.000001 in
	rotary	0.00001 deg	0.00001 deg
ISE	linear	0.000001 mm	0.0000001 in
	rotary	0.000001 deg	0.000001 deg

Any axis can be designated for data input and position display in ***diameter*** by setting DIA=1 in the parameter N0106 Axis Properties.

3 Preparatory Functions (Codes G)

The nature of an instruction in a given block is determined by the address G and the number following it.
There are so-called ***one-shot codes G*** effect of which is valid in the given block, and there are so-called ***modal functions G*** effect of which is valid until this effect is switched off or changed by another code G.
The one-shot codes G belong to the group numbered with 0 (zero). Those ones from the modal codes G which influence each other have ***group number*** different from 0 (zero).
More than one code G may be written into one block provided that only one of the functions belonging to the same group has to be per group.
The ***leading zero has not to be written in the code*** but the control accepts it. For example: Either G01 or G1 may be written in the program.
The table below contains codes G interpreted by the control, their group numbers and functions.

Code G	**Group**	**Function**	**Page**
G0*	1	Positioning	25
G1*		Linear Interpolation	27
G2		Circular and spiral (planar, helical and conical) interpolation, clockwise (CW)	29, 34, 36, 39
G3		Circular and helical (planar, cylindrical and conical) interpolation, counter-clockwise (CCW)	29, 34, 36, 39
G4	0	Dwell	78
G9		Exact stop in the given block	65
G7.1		Cylindrical interpolation	48
G10		Programmed data input	89, 90, 128,
G11		Programmed data input cancel	
G10.9		Interchange radius/diameter programming from program	53
G12.1	21	Polar coordinate interpolation on	43
G13.1*		Polar coordinate interpolation off	43
G15*	17	Data specification in polar coordinates off	54
G16		Data specification in polar coordinates on	54

Code G	Group	Function	Page
G17*	02	Selection of the plane X_pY_p	94
G18*		Selection of the plane Z_pX_p	94
G19*		Selection of the plane Y_pZ_p	94
G20*	6	Data input in inch	52
G21*		Data input in mm	52
G22*	04	Programable stroke check function on	283
G23*		Programable stroke check function off	283
G28	0	Programmed return to the reference point	80
G30		Return to the second, third and fourth reference point	81
G31		Skip function	277
G33	01	Thread cutting	41
G36	0	Automatic tool length measurement in the direction X	280
G37		Automatic tool length measurement in the direction Z	280
G38		Holding the tool nose radius compensation vector	158
G39		Corner rounding with tool nose radius compensation	159
G40*	07	Calculation of tool nose radius compensation cancel	133
G41		Calculation of tool nose radius compensation from the left	133
G42		Calculation of tool nose radius compensation from the right	133
G43.7*	08	Tool Length Compensation by Code G	131
G49*		Tool length compensation off	131
G50*	11	Scaling off	171
G51		Scaling on	171
G50.1*	22	Mirroring off	174
G51.1		Mirroring on	174
G51.2	31	Polygonal turning on	256
G50.2*		Polygonal turning off	256
G52	0	Coordinate offset	91
G53		Positioning in the machine coordinate system	84

Code G	Group	Function	Page
G54*	14	Selection of the workpiece coordinate system 1	86
G55		Selection of the workpiece coordinate system 2	86
G56		Selection of the workpiece coordinate system 3	86
G57		Selection of the workpiece coordinate system 4	86
G58		Selection of the workpiece coordinate system 5	86
G59		Selection of the workpiece coordinate system 6	86
G54.1		Selection of additional workpiece coordinate system	87
G61	15	Exact stop mode	65
G62		Automatic corner override mode	65
G63		Override inhibition	65
G64*		Continuous cutting mode	65
G65	0	Simple macro call	339
G66	12	Modal macro call after each motion command	341
G66.1		Modal macro call from each block	342
G67*		Modal macro call cancel	341, 343
G68	16	In-plane rotation of an object about a given point	169
G69*		In-plane rotation off	169
G70	0	Finishing cycle	217
G71		Stock removal cycle in turning	197
G72		Stock removal cycle in facing	208
G73		Pattern repeating cycle	214
G74		Face grooving cycle	218
G75		Grooving cycle	220
G76		Thread cutting cycle	222
G77	1	Turning cycle	190
G78		Simple thread cutting cycle	192
G79		Facing cycle	194

Code G	Group	Function	Page
G83.1	09	High-speed peck drilling cycle	235
G84.1		Left-handed tapping cycle using spring tap	236
G86.1		Boring Cycle with Automatic Tool Shift	237
G80*		Cycle state cancel	238
G81		Drilling cycle, rapid-traverse retraction	238
G82		Drilling cycle with dwell, rapid-traverse retraction	239
G83		Peck drilling cycle	240
G84		Right-handed tapping cycle using spring tap	241
G84.2		Right-handed rigid tapping cycle Right-handed peck rigid tapping cycle	242 244
G84.3		Left-handed rigid tapping cycle Left-handed peck rigid tapping cycle	242 244
G85		Boring cycle, retraction with feed	247
G86		Boring Cycle with Retraction with Spindle in Standstill	248
G87		Manual Control/Back Boring Cycle	249
G88		Boring cycle, manual operation at the bottom point	251
G89		Boring cycle, dwell at the bottom point, retraction with feed	252
G80.8	34	Electronic gear box off	259
G81.8		Electronic gear box on	259
G90*	03	Absolute dimensioning	51
G91*		Incremental dimensioning	51
G92	00	Creating a new work coordinate system	90
G94*	05	Feed per minute	64
G95*		Feed per revolution	64
G96	13	Calculation of constant cutting speed on	100
G97*		Calculation of constant cutting speed off	100
G98*	10	Return from the drilling cycle to the initial point	218
G99		Return from the drilling cycle to the (approach) point R	218

Basic State after Turning on

Codes G marked with * in a group represent state the control takes on ***after turning on.*** Where there are several codes G marked with * in a group, on the basis of the parameters N1300 DefaultG1 and N1301 DefaultG2 can it be selected which ones have to be valid ***after turning on.*** These codes G are as follows:

G00, G01;
G17, G18, G19;
G20, G21;
G22, G23;
G90, G91;
G94, G95.

Basic State after Pushing Reset Key or after Program End

When ***reset*** key is pushed or a ***program ends*** (M2, M30), if the bit CLR of the parameter N1301 DefaultG2
= 0: the control, without any condition, will take on state marked with * in the G code table, or it will reset the code G values into after-turning-on basic state set in the parameters N1300 DefaultG1 and N1301 DefaultG2;
= 1: the control, on the basis of values given in the parameter CLR G Table1, 2, 3, 4, 5, will get into basic state according to the G code groups, or it will leave the modal values unchanged.
If in the parameter CLR G Table1, 2, 3, 4, 5 the G code group's bit Cnn (where nn is the group number of the code G) is
= 0: the control will set the adequate G code group into basic state;
= 1: the control will leave the adequate G code group in emerged and inherited state.

4 Interpolation

4.1 Positioning (G0)

The positioning command G0 moves the tool along all the axes programmed in the block to the specified point.
Motion is performed using rapid traverse. The value of rapid traverse is specified by the builder of the machine tool in parameter for each axis, and it cannot be set from the program.
In the case of absolute dimensioning, the tool moves to the point of given coordinates in the coordinate system of the actual workpiece.
In the case of incremental dimensioning , the tool moves the given distance from its actual position.
The format of the block is:

G0 v

where v are the coordinates given in the block. The designation v refers here (and hereafter) to all the controlled axes used in the given channel. Simultaneous positioning along all the axes of the channel is possible.
Instead of G0, G00 can be given too.

An example:
```
G0 X150 Z2
```
Other code G and function can also be given in the block.

An example:
```
G0 G90 X150 Z2 S2000 M3
```

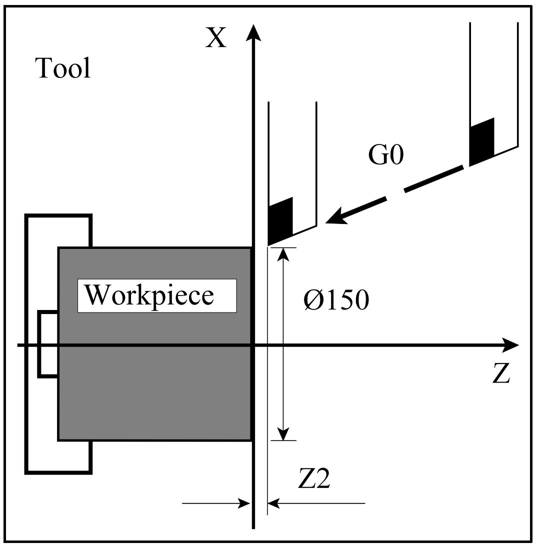


Fig. 4.1-1

The modal code G0 remains effective until it is re-written by another interpolation command.
An example:
```
G0 X150 Z2
X20 Z1 (positioning block, because G0 is being modal)
```
After turning on code G0 will be in effect if bit position of the parameter N1300 DefaultG1 is G01=1.

4.1.1 Positioning by Linear Interpolation

In the case of moving several axes simultaneously, the control will move the tool along a straight line connecting the starting point and the end point while positioning if bit position of the parameter N0421 Acc Contr is ROL=0.
The control calculates the resultant velocity vector (**v**) in such a way that positioning be executed in a minimum time, and the value of velocity along none of the axes does not exceed the rapid traverse value set for the given axis.
After completing motion, the control will check the 'in-position' signal if bit position of the parameter N1337

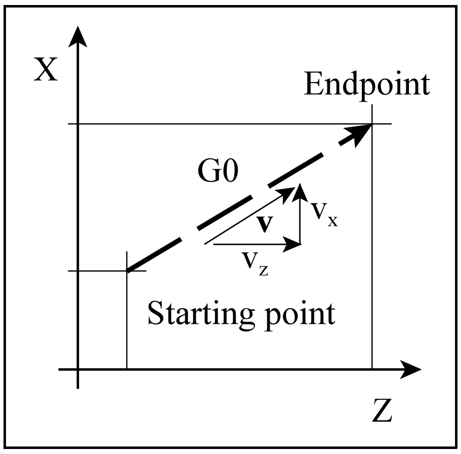


Fig. 4.1.1-1

Execution Config is PCH=1.
The control waits for the ‘in-position’ signal for the time set in the parameter N1340 Inpos Timeout ; if the signal does not received even from this time on, the control will send the error message ‘*2501 Position error*’.
The maximum acceptable deviation from the position can be specified in the parameter N0516 Inpos, for each axis.
In-position check should be set when it is reasonable, otherwise the execution time could increase. For example, executing the program detail

```
        G0 X60
        Z1
        X56
```

the difference between the two time periods will be as it is illustrated in the figure.

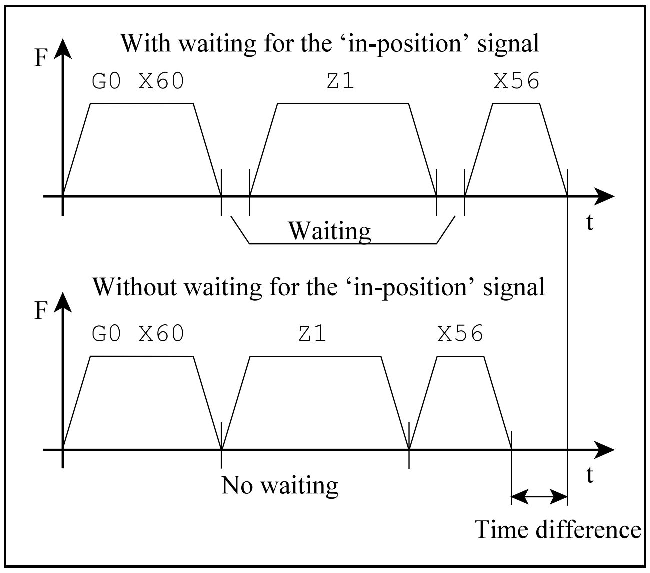

With waiting for the ‘in-position’ signal
F
G0 X60
Z1
X56
Waiting
t
Without waiting for the ‘in-position’ signal
F
G0 X60
Z1
X56
No waiting
t
Time difference

Fig. 4.1.1-2

The control ***always carries out in-position check*** at the end of the block

in state G61 (exact stop mode), or in the positioning block in which code G9 (exact stop) has been written, even at parameter position PCH=0 (there is no in-position check).

4.1.2 Positioning by Overlapping the Rapid Traverse Motions

Consecutive positionings carried out on different axes can be accelerated further by the use of overlapping the motion of the positioning blocks. This means, that while in a positioning block one of the axes decelerates approaching the end point position, an other axis being in the next positioning block begins accelerate already.
Overlapping will be switched on by the bit position ROL=1 of the parameter N0421 Acc Contr.
The percentage of speed, after reaching which in the deceleration period of the previous block the next block starts, can be set in the parameter N0422 Rapid Reduct. Ratio, in percentage.

Using the former example, executing the program detail

```
        G0 X60
```

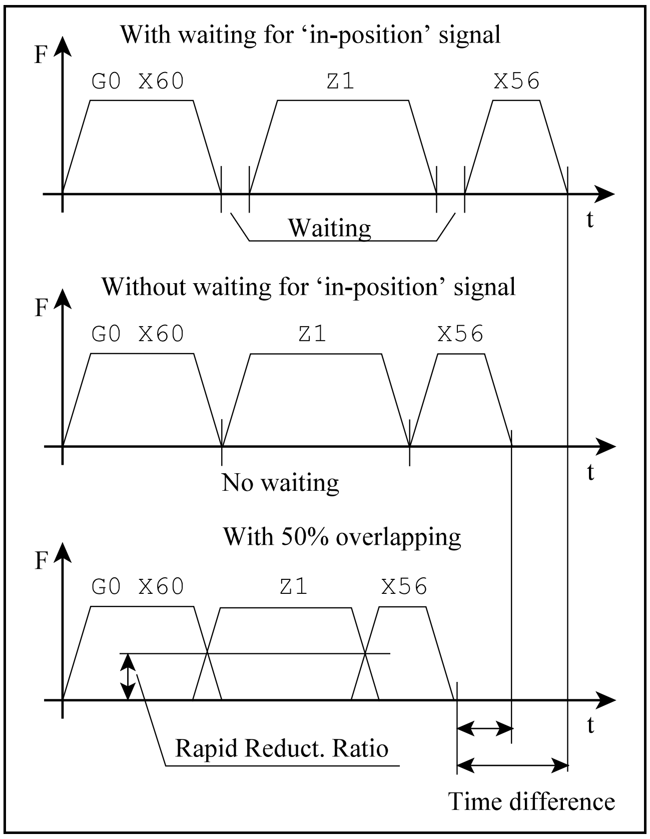


Fig. 4.1.2-1

```
Z1
X56
```

the difference between execution times will be as it is illustrated in the figure.

In the case of positioning with overlap, tool path is not cornered, but it is a rounded one. Because of this, care should be taken in retracting the tool from the workpiece; maybe degree of retracting must be higher than usual in the program.

In the case of positioning ***several axes*** programmed in one block, the ***tool motion path is straight approximately only***, and the different axes arrive to the position with time difference.

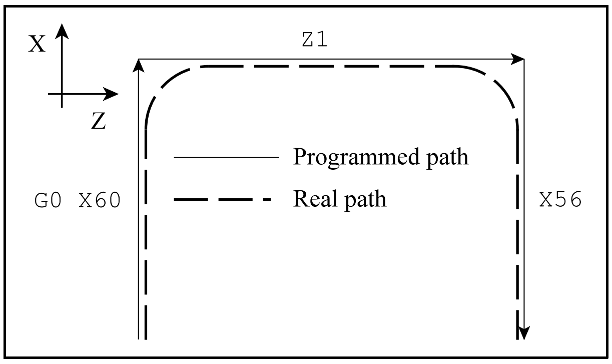


Fig. 4.1.2-2

The control ***suspends overlapping*** between positioning blocks and ***always carries out in-position check*** at the end of the block

in state G61 (exact stop mode), or in the positioning block in which

code G9 (exact stop) has been written, even at parameter position PCH=0 (there is no in-position check)

4.2 Linear Interpolation (G1)

The linear interpolation command G1 moves the tool to the specified point along straight path, along all the axes programmed in the block.
Motion is carried out at feed F programmed in block, or being modal.
In the case of absolute data specification, the tool moves in the actual workpiece coordinate system to the point of specified position.
In the case of incremental data specification, the tool steps the specified distance from its actual position.

The format of the block is:

G1 v F

where v are the coordinates given in the block, and F is the value of the feed. Motion along all the axes of the channel at the same time is possible.
Instead of G1, G01 can be given too.

An example:

```
G1 X136 Z26 F0.5
```

In the block, other code G and function can also be given.
Példa:

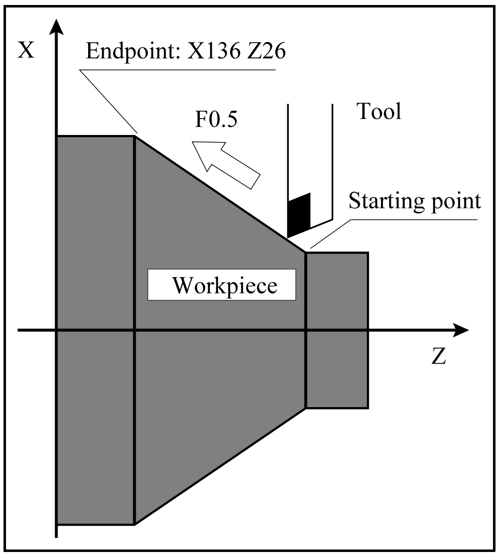


Fig. 4.2-1

```
        G1 G91 X5 Z10 S2000 M3
```

The modal code G1 remains effective until it is re-written by another interpolation command. An example:

```
        G1 X136 Z26 F0.5
        Z0 (linear interpolation by F.5, because G1 and F are being
        modal)
```

After turning on code G1 will be in effect if bit position of the parameter N1300 DefaultG1 is G01=1.

The feed programmed at the address F is valid always along the path programmed. Its axial components are as follows:

Feed along the axis X is: $F_x = \frac{\Delta x}{L} F$

Feed along the axis Z is: $F_z = \frac{\Delta z}{L} F$

....

The formula continues for all the axes programmed in the block.

where: Δx, Δz, ...: displacements calculated along the respective axes,

L: length of the programmed displacement:

$$L = \sqrt{\Delta x^2 + \Delta z^2 + \ldots}$$

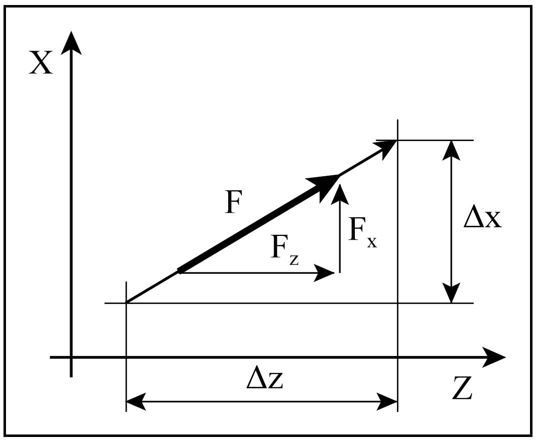


Fig. 4.2-2

The feed along a rotational axis is interpreted in unit of degrees per minute (°/min). In the block

```
        G1 C270 F120
```

F120 means: 120 deg/min.

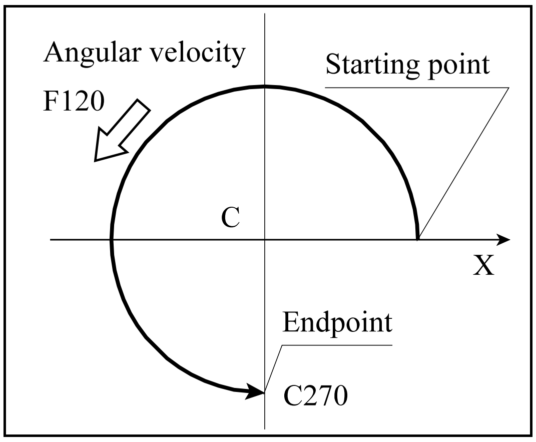


Fig. 4.2-3

If motion of a linear axis and a rotary axis is combined through linear interpolation, the feed components will be distributed according to the above formulas.

For example, in the block:

```
        G91 G01 Z100 C45 F120
```

the feed components in the Z and B directions are as follows:

Feed along the axis Z is: $F_z = \frac{100}{\sqrt{100^2 + 45^2}} 120 = 109.4$ mm/min

Feed along the axis C is: $F_c = \frac{45}{\sqrt{100^2 + 45^2}} = 49.2$ °/min

4.3 Circular Interpolation (G2, G3)

The G2 or G3 command moves the tool in the selected plane to the point specified in the block, along circular arc. Motion is carried out at feed F programmed in block, or being modal.
The format of the block is:

$$G17\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} X_p\_\ Y_p\_ \begin{Bmatrix} R\_ \\ I\_\ J\_ \end{Bmatrix} F\_$$

$$G18\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} X_p\_\ Z_p\_ \begin{Bmatrix} R\_ \\ I\_\ K\_ \end{Bmatrix} F\_$$

$$G19\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} Y_p\_\ Z_p\_ \begin{Bmatrix} R\_ \\ J\_\ K\_ \end{Bmatrix} F\_$$

Circular interpolation is accomplished in the plane selected by the commands G17, G18, G19; in the case of ***G2*** in ***clockwise*** direction, and in the case of G3 in ***counter-clockwise*** direction:

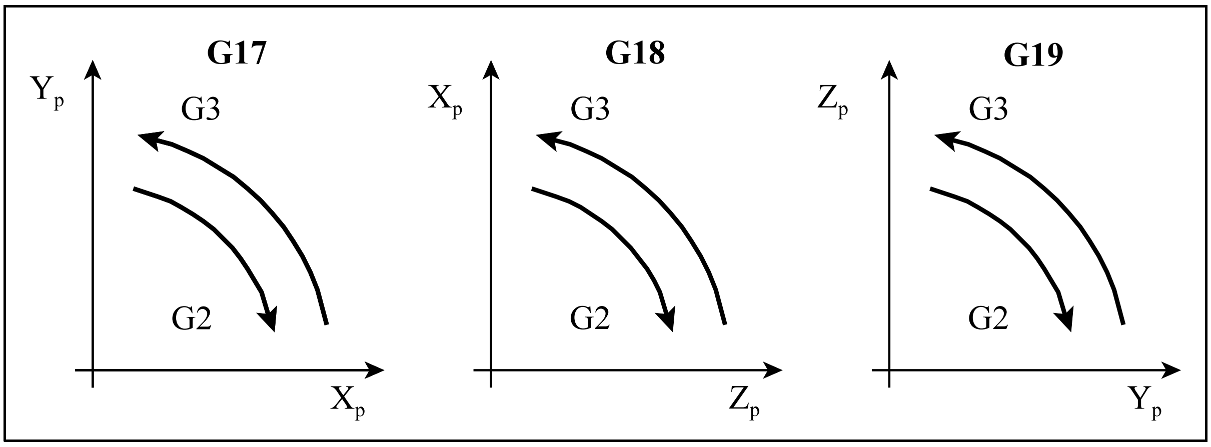


Fig. 4.3-1

Instead of codes G2 and G3, codes G02 and G03 can also be written in the program.
The codes G2 and G3 are modal ones, they remain effective until they are re-written by another interpolation command.

Here and hereafter, the meanings of X_p, Y_p, and Z_p are as follows:

X_p: axis X or an axis parallel with it;
Y_p: axis Y or an axis parallel with it;
Z_p: axis Z or an axis parallel with it.

The values of ***X_p, Y_p, and Z_p*** are ***the coordinates of the circle endpoint*** in the given coordinate system, specified as absolute data, or incremental data measured from the starting point.
Further data of the circle can be specified in two different ways:

Case 1: Specifying the circle by its radius at the address R

In this case, the control automatically calculates the coordinates of the circle center from the coordinates of the starting point (this is the point where the control is in the moment of reading-in the block of the circle), from the coordinates of the end point (values defined at the addresses X_p, Y_p and Z_p) and from the programmed circle radius R. In the case of a given circulation direction (G2 or G3), two different circles of radius R can be drawn passing the starting and end points.

If the circle radius ***R*** is given

as a ***positive*** number, the control will interpolate an arc ***smaller than 180°***,

as a ***negative*** number, the control will interpolate an arc ***larger than 180°***. For example:

Arc section 1: `G2 X80 Z50 R40`
Arc section 2: `G2 X80 Z50 R-40`
Arc section 3: `G3 X80 Z50 R40`
Arc section 4: `G3 X80 Z50 R-40`

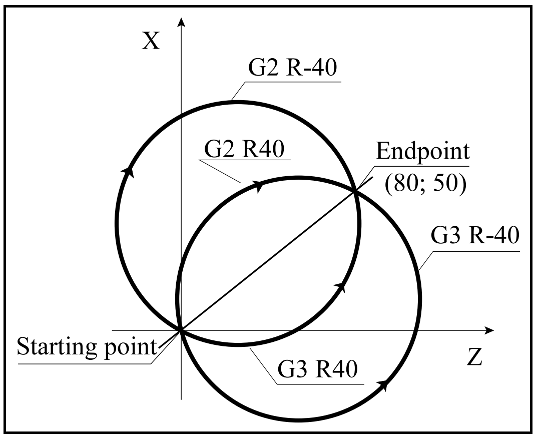


Fig. 4.3-2

Case 2: Specifying the circle by its center at the addresses I, J and K

The control interprets the values specified at the addresses I, J and K ***incrementally*** in such a way that the vector defined by the values I, J and K points ***from the starting point to the center point*** of the circle.

The values ***I, J and K*** must always be specified in ***radius***, even if the axes adherent to them are set to be programmed in diameter.

For example:

in the case of G17: `G3 X20 Y140 I-50 J-20 (X, Y are specified in diameter)`

in the case of G18: `G3 X140 Z10 I-20 K-50 (X is specified in diameter)`

in the case of G19: `G3 Y20 Z70 J-50 K-20 (Y is specified in diameter)`

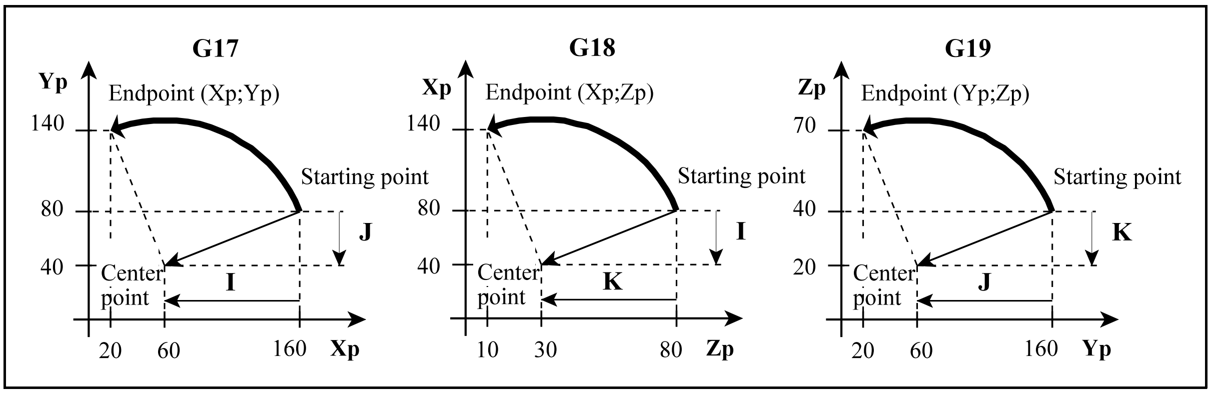


Fig. 4.3-3

At the address F, the feed along the path can be programmed which points always in the direction of the circle tangent and is constant along the whole path.

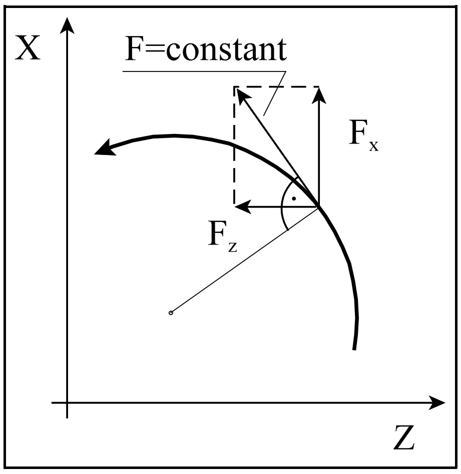


Fig. 4.3-4

An example:
Programming the path shown in the figure.

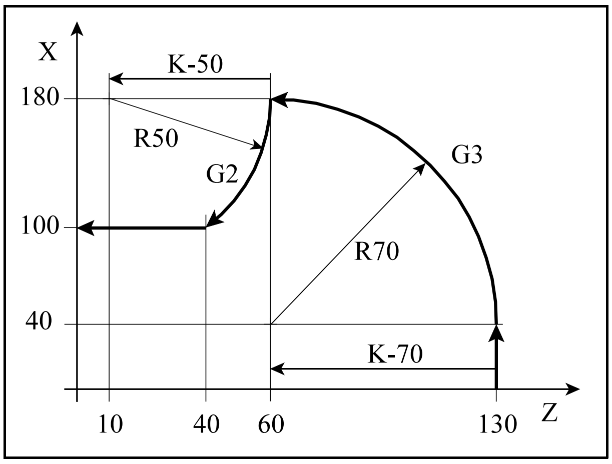


Fig. 4.3-5

Programming a circle using absolute coordinates and specifying R:

```
G90 G18
G0 X0 Z130 M3 S1000
G1 X40 F500
G3 X180 Z60 R70
G2 X100 Z40 R50
G1 Z0
...
```

Programming a circle using absolute coordinates and specifying the circle center I, K:

```
G90 G18
G0 X0 Z130 M3 S1000
G1 X40 F500
G3 X180 Z60 K-70
G2 X100 Z40 K-50
G1 Z0
...
```

Programming a circle using incremental coordinates and specifying R:

```
G90 G18 G0 X0 Z130 M3 S1000
G91
G1 X40 F500
G3 X140 Z-70 R70
G2 X-80 Z-20 R50
G1 Z-40
...
```

Programming a circle using incremental coordinates and specifying the circle center I, K:

```
G90 G18 G0 X0 Z130 M3 S1000
G91
G1 X40 F500
G3 X140 Z-70 K-70
G2 X-80 Z-20 K-50
G1 Z-40
...
```

Specification of ***I0, J0*** and ***K0*** may be omitted. For example:

```
G0 X0 Z100 F500
G18 G03 X200 Z0 K-100
```

In the case of programming a quarter circle with radius of 100 mm and with center in the origin, I0 has not to be written, since the distance of the circle center in the direction X from the point X0 Z100 is 0.

If all three of X_p, Y_p and Z_p are omitted, then

– if coordinates of circle center are specified at the addresses I, J and K, the control will interpolate a full circle with the arc of 360°. For example, in the case of

```
G0 X400 Y0 F500
G17 G03 I-100
```

the control interpolates a full circle with radius 100 mm and with center point of X200 Y0;

– if radius R is specified, for example

```
G0 X0 Z200 F500
G18 G03 R100
```

the control will not move and will not indicate error.

If the circle block contains ***neither radius R nor I, J and K***, the control will send the error message *'2015 Circle definition error'*.

If reference to the addresses ***I, J and K being outside of the selected plane*** is made, the control will send the error message '*2015 Circle definition error'*.

If the ***difference between the starting point radius and the end point radius of the circle*** specified in the G2, G3 block is greater than the value given in the parameter *N1339 Radius Diff*, the control will send the error message '*2012 Circle radius difference*'.

If the difference between the radii less than the value given in the abovementioned parameter, the control will move the tool along a planar spiral path where the radius changes linearly as a function of the central angle.

In the case of interpolation of an arc with variable radius, not the tangential velocity, but the angular one will be constant.

The value of the parameter N1339 Radius Diff must be greater than 0, for example 0.001, otherwise the control will send unnecessary error indications.

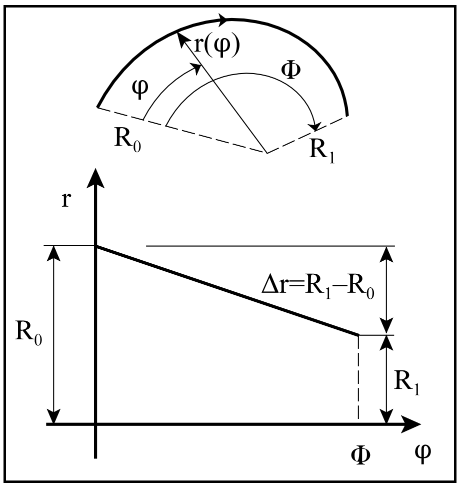


Fig. 4.3-6

Radius difference error or interpolation of a circle of variable radius may be occurred in the following cases:

When the circle center positioning specified at the addresses I, J and K is not correct. For example:

```
     G18 G90 G0 X0 Z50
     G3 Z-20 K-50
```

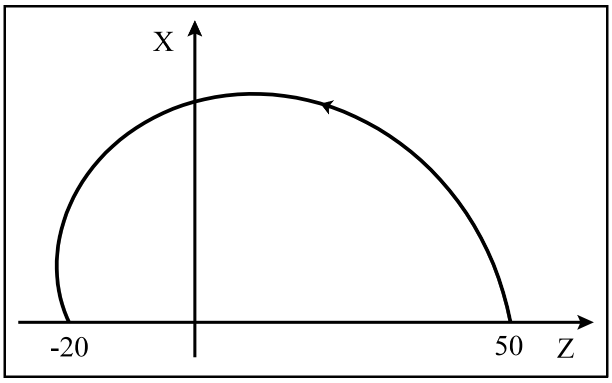


Fig. 4.3-7

If the given circle radius is smaller than the half of the straight line segment between the starting and end points, the control will regard the given circle radius as the starting-point-radius, and it will interpolate such a circle of variable radius, the center of which is on the line linking the starting and end points at a distance R from the starting point:

```
     G18 G0 G90 X0 Z0
     G2 X60 Z40 R10
```

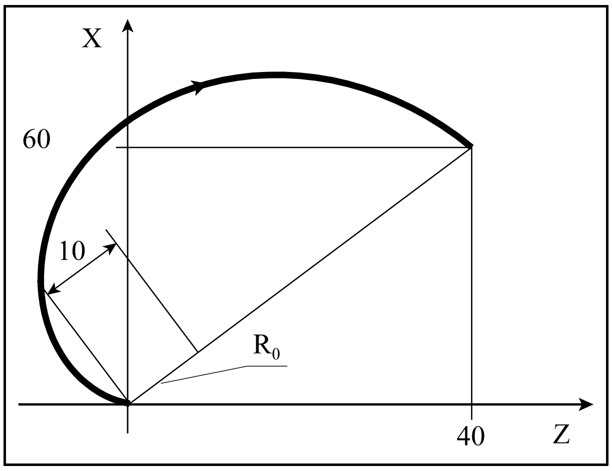


Fig. 4.3-8

The circle center position, i.e, I, J and K can also be given by absolute value calculated from the workpiece zero point. For this, the set position #2 CCA=1 of the parameter N1337 Execution Config is necessary. This case should be avoided.

4.3.1 Planar Spiral Interpolation (G2, G3)

By the command G2 or G3, planar spiral interpolation can be programmed in the way of giving the number of revolutions of the spiral too, at the address L. ***The center and the end point of the circle*** have to be specified in such a way ***that the starting-point-radius and the end-point-radius be different***. Motion will be executed at a feed F programmed in the block or being modal.
The format of the block is:

$$G17\begin{Bmatrix}G2\\G3\end{Bmatrix}X_p\_Y_p\_I\_J\_L\_F\_$$

$$G18\begin{Bmatrix}G2\\G3\end{Bmatrix}X_p\_Z_p\_I\_K\_L\_F\_$$

$$G19\begin{Bmatrix}G2\\G3\end{Bmatrix}Y_p\_Z_p\_J\_K\_L\_F\_$$

The values ***X_p, Y_p, Z_p*** are the ***coordinates of the spiral end point*** in the given coordinate system; they are specified as absolute data, or incremental data measured from the starting point.
At the addresses ***I, J and K***, ***the coordinates of the center point of the spiral*** are specified as ***incremental*** data measured from the starting point in such a way that the vector defined by the values I, J and K points ***from the spiral starting point to the spiral center***.
The values ***I, J and K*** will have to always be given ***in radius***, even if the axes adherent to them are set to be programmed in diameter.
The number of revolutions of the spiral is given ***at the address L.*** Every start will mean a new revolution even if the next one is not a full revolution. The address ***L*** is a ***positive integer number***.

In the course of spiral interpolation, the control varies the starting-point-radius (R_0) with the swivel angle (φ) linearly in such a way that the end-point-radius correspond to the programmed data, when the radius executed revolutions specified at the address L and reached the end point position.
It follows from the foregoing that in the course of spiral interpolation what is indicated for the control by filling in the address L, the starting-point-radius differs from the end-point-radius. If the address L is filled in, the control will never check the maximum value of the radius difference specified in the parameter N1339 Radius Diff.
The ***feed F*** specified in the spiral interpolation remains ***constant along the full length of the spiral***.

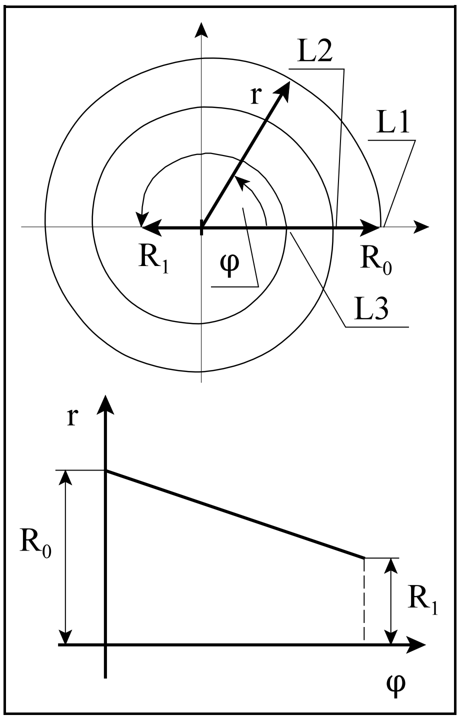


Fig. 4.3.1-1

An example:
Programming the spiral shown in the figure. The starting point of the spiral is X180 Y0, its center measured from the starting point is I-90, J0, the radius variation per revolution is 24, and the number of revolutions is 2.5. So, the end-point radius of the spiral is:

R=90-2*24-24/2=30

The number of revolutions entered is 3.
The program is as follows:

```
G17 G90 G94 M3 S1000
G0 X180 Y0 F1000
G3 X-60 I-90 L3
```

In the above example, the coordinates ***X*** and ***Y*** is specified in ***diameter***.

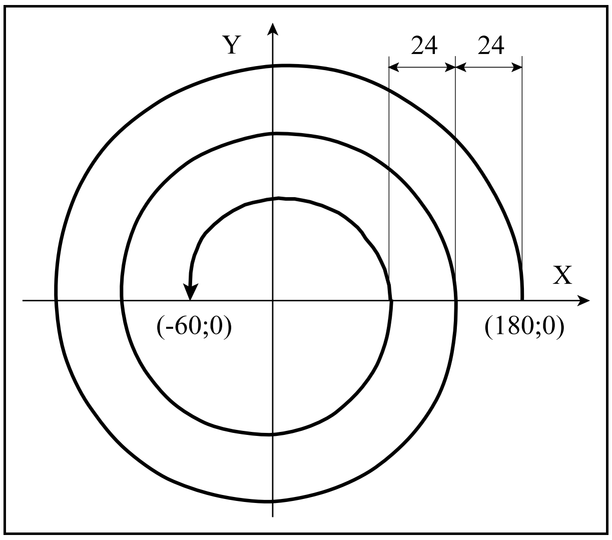


Fig. 4.3.1-2

If we want the control not to indicate error in the loop 2012 Circle radius difference as it is illustrated by the example given in the subsection 4.3 Circular interpolation (page 29), the program will have to be modified in the following way:

```
G18 G90 G0 X0 Z50
G3 Z-20 K-50 L1
```

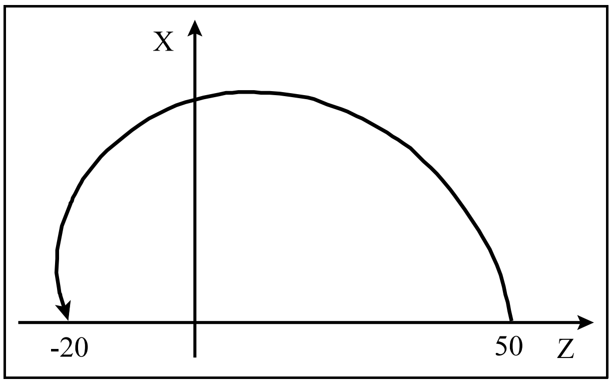


Fig. 4.3.1-3

4.3.2 Helical Spiral Interpolation (G2, G3)

By the command G2 or G3, helical spiral interpolation can be programmed in the way of ***programming motion along the axis perpendicular to the plane of the circle***. ***The number of revolution of the spiral*** can be given ***at the address L***. Motion will be executed at a feed F programmed in the block or being modal.
The format of the block is:

$$G17\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} X_P\_ Y_P\_ \begin{Bmatrix} R \\ I\_J\_ \end{Bmatrix} Z_P\_ q\ldots L\_ F\_$$

$$G18\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} X_P\_ Z_P\_ \begin{Bmatrix} R\_ \\ I\_K\_ \end{Bmatrix} Y_P\_ q\ldots L\_ F\_$$

$$G19\begin{Bmatrix} G2 \\ G3 \end{Bmatrix} Y_P\_ Z_P\_ \begin{Bmatrix} R\_ \\ J\_K\_ \end{Bmatrix} X_P\_ q\ldots L\_ F\_$$

Specification of the circle is carried out according to the rules determined for circle interpolation. Displacement along the axis perpendicular to the plane of the circle arc is proportional to displacement along the circle arc.
In addition to the axis perpendicular to the circle, displacement can be specified for axes of arbitrary quantity, which are marked with q in the formula above and programmable in the channel.
The number of revolutions of the helical spiral is given ***at the address L.*** Every start will mean a new revolution even if the next one is not a full revolution. The address ***L*** is a ***positive integer number***.
In the course of helical spiral interpolation, the control varies the pitch with the swivel angle (φ) linearly in such a way that the endpoint-position on the axes which are outside of the plane of the circle correspond to the programmed data, when revolutions specified at the address L have been executed and the end point position have been reached.

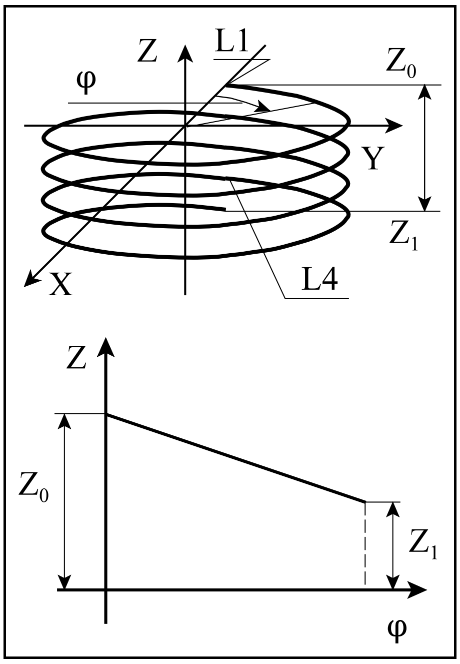


Fig. 4.3.2-1

Examples:
The helical spiral interpolation shown in the figure can be specified in the following way:

```
      G17 G90 G0 X100 Y0 Z0
      G03 X0 Y100 Z20 R50 F150
```

In this example, the coordinates ***X*** and ***Y*** are specified ***in diameter***.

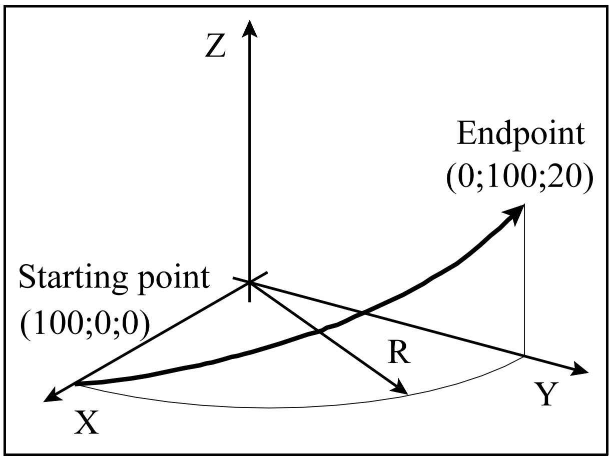


Fig. 4.3.2-2

In this figure, milling of a circle arc into the mantle of an oblique cylinder is illustrated. The axis V is parallel with the axis Y and is moved together with the axis Z by the control:

```
      G17 G90 G0 X100 Y0 Z0 V0
      G03 X0 Y-100 Z50 V20 I-50
```

In this example, the coordinates ***X*** and ***Y*** are specified ***in diameter***.

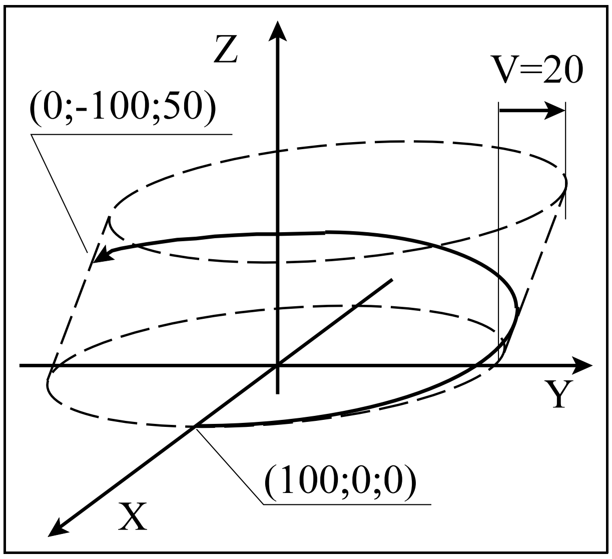


Fig. 4.3.2-3

This picture is for programming a helical spiral with 4 revolutions.
The starting point of the spiral is X-100 Y0 and Z0, its center measured from the starting point is I-50, J0, the pitch per revolution is 5, and the number of full revolutions is 4.
The program is:

```
      G54 G17 G90 ...
      G0 X-100 Y0
      Z0
      G2 I50 Z-20 L4 F100
      ...
```

In this example, the coordinates ***X*** and ***Y*** are specified ***in diameter***.

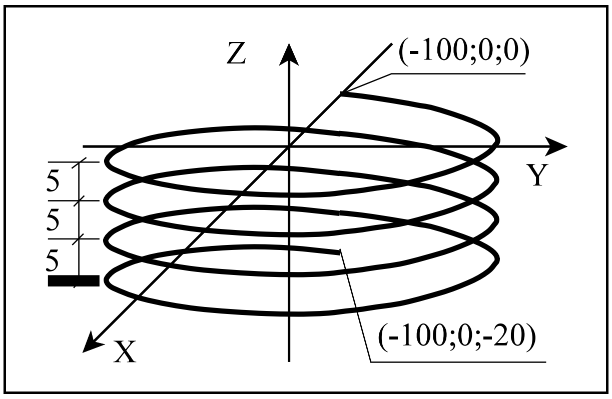


Fig. 4.3.2-4

Basically, the feed given at the address ***F*** is valid ***along circle path*** when the bit position of the parameter N1337 Execution Config #3 is ***HEF=0***. Then, the following formulas give feed for the axis:

$$F_{arc} = F$$

$$F_q = \frac{L_q}{L_{arc}} F$$

In the case of bit position ***HEF=1***, feed will be calculated by the control along spiral path. Then, the following formulas give feed for the axis:

$$F_{arc} = \frac{L_{arc}}{\sqrt{L_{arc}^2 + L_q^2}} F$$

$$F_q = \frac{L_q}{\sqrt{L_{arc}^2 + L_q^2}} F$$

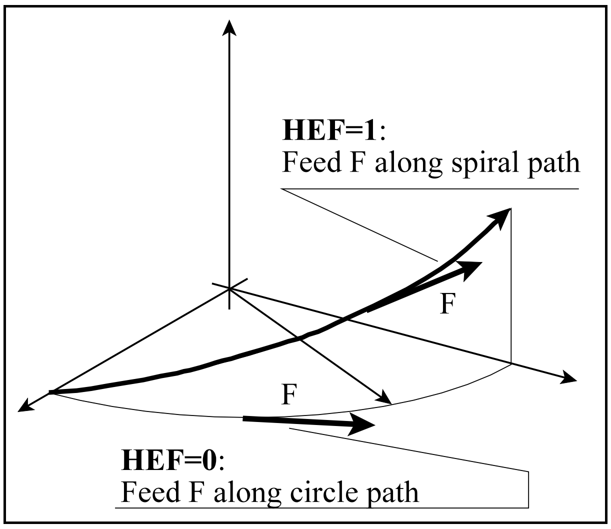


Fig. 4.3.2-5

where	L_q:	the displacement along the axis q;
	L_{arc}:	the length of the circle arc;
	F:	the programmed feed;
	F_q:	the feed along the axis q;
	F_{arc}:	the feed along the circle arc.

The specified ***tool radius compensation*** is always valid in the plane of the ***circle*** along the circle path.
In that case, if ***the radius of the circle*** given in the selected plane ***varies***, interpolation will be executed along the mantle of the given cone.

4.3.3 Conical Spiral Interpolation (G2, G3)

By the command G2 or G3, conical spiral interpolation can be programmed in the way of ***programming motion along the axis perpendicular to the plane of the circle***. ***The center point and the end point of the circle*** have to be specified in such a way that ***the starting-point-radius*** and ***the endpoint-radius*** of the circle have to be ***different***. ***The number of revolution of the conical spiral*** can be given ***at the address L***. Motion will be executed at a feed F programmed in the block or being modal.
The format of the block is:

$$G17\begin{Bmatrix}G2\\G3\end{Bmatrix}X_P\_Y_P\_I\_J\_Z_P\_q\dots L\_F\_$$

$$G18\begin{Bmatrix}G2\\G3\end{Bmatrix}X_p\_Z_p\_I\_K\_Y_P\_q\dots L\_F\_$$

$$G19\begin{Bmatrix}G2\\G3\end{Bmatrix}Y_p\_Z_p\_J\_K\_X_P\_q\dots L\_F\_$$

The values ***X_p, Y_p, Z_p*** are the ***coordinates of the spiral end point*** in the given coordinate system; they are specified as absolute data, or incremental data measured from the starting point.
In addition to the axis perpendicular to the circle, displacement can be specified for axes of arbitrary quantity, which are marked with q in the formula above and programmable in the channel.
At the addresses ***I, J and K***, ***the coordinates of the center point of the conical spiral*** are specified in the selected plane as ***incremental*** data measured from the starting point in such a way that the vector defined by the values I, J and K points ***from the spiral starting point to the spiral center***.
The values ***I, J and K*** will have to always be given ***in radius***, even if the axes adherent to them are set to be programmed in diameter.
The number of revolutions of the spiral is given ***at the address L.*** Every start will mean a new revolution even if the next one is not a full revolution. The address ***L*** is a ***positive integer number***.

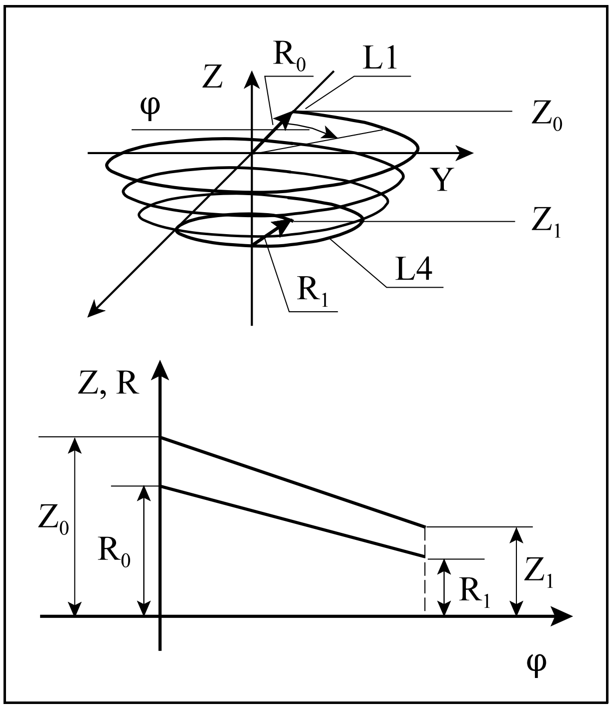


Fig. 4.3.3-1

In the course of conical spiral interpolation, the control varies the pitch with the swivel angle (φ) linearly in such a way that the endpoint-position on the axes which are outside of the plane of the circle corresponds to the programmed data, when revolutions specified at the address L have been executed and the end point position have been reached. It will also vary linearly the circle radius with the swivel angle (φ).

An example:
This picture is for programming a 4-revolution conical spiral whose starting-point-radius is 50, endpoint-radius is 20, and pitch per revolution is 5.
The program is:

```
        G17 G90 ...
        G0 X-100 Y0
        Z0
        G2 X-40 I50 Z-20 L4 F100
        ...
```

In this example, the coordinates ***X*** and ***Y*** are specified ***in diameter***.

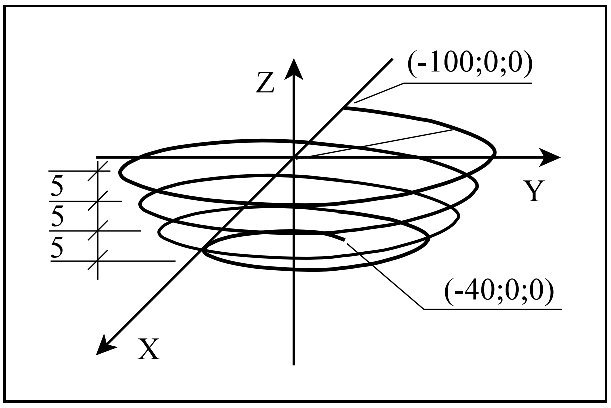


Fig. 4.3.3-2

4.4 Equal Lead Thread Cutting (G33)

By the instruction G33, cutting of straight or tapered thread of equal lead can be programmed. The format of the block is:

G33 v F Q

or

G33 v E Q

For vector ***v***, coordinate data of ***maximum two axes*** can be written. If data of two coordinates are assigned for the vector v, the control will cut tapered thread. Lead will be taken into account by the control on the axis along which the displacement is longer, i.e.
if ***α<45°***, i.e. Z>X, the programmed lead will be taken into account ***along the axis Z***,
if ***α>45°*** , i.e. X>Z, the programmed lead will be taken into account ***along the axis X***.
The lead can be defined in the following two ways:

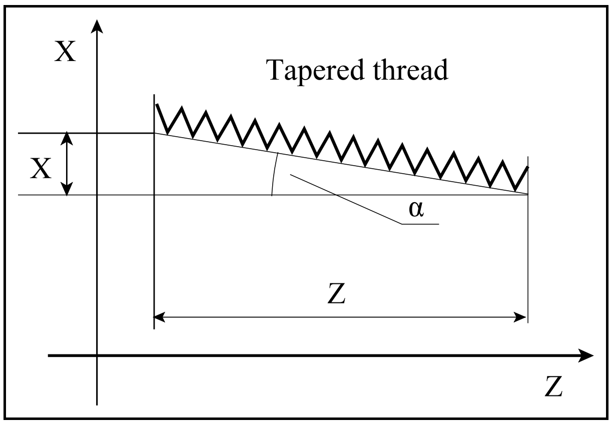


Fig. 4.4-1

Specification of the lead at address F
If the lead is specified at ***address F***, the data will be interpreted in ***mm/rev*** or ***inch/rev.*** Accordingly, F2.5 will has to be programmed if a thread of 2.5 mm lead is to be cut.

Specification of the lead at address E
If the lead is specified at address E, the control will cut inch thread. Interpretation of address E is number of threads per inch. If E8 is programmed, the control will cut a thread, characterized by the lead of 1/8"=25.4/8=3.175 mm.

The ***value of angle*** in degree by which ***the spindle has to rotate from the zero pulse of the spindle encoder*** before starting the thread cutting can be given ***at the address Q***. ***Multiple-start thread*** can be cut by an adequate programming of the value of ***Q***, i.e. the control can be programmed here for the particular angular displacements of the spindle, at which cutting the various starts of the thread are to be started. For example, if a double-start thread is to be cut, the first start will has to be commenced from Q0 (no special programming is needed), while the second start from the Q180.

G33 is a modal function. If several thread-cutting blocks are programmed in succession, it is possible to cut thread on an arbitrary surface bounded by straight line segments.

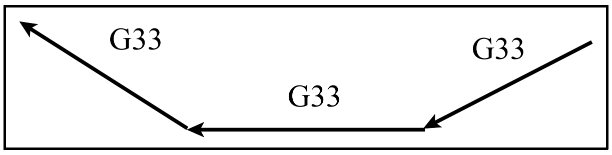


Fig. 4.4-2

The control is synchronized to the zero pulse of the spindle encoder in the first block, no synchronization will be performed in the subsequent blocks what results in a continuous lead in each line segment. Therefore, the programmed angular displacement Q of the spindle will also be taken into account in the first block only.

An example:
Cutting a thread with the lead of 2 mm:

```
...
G0 G90 X50 Z60
X20
G33 X30 Z12 F2
G0 X50
Z60
...
```

In the example above X is specified in diameter.

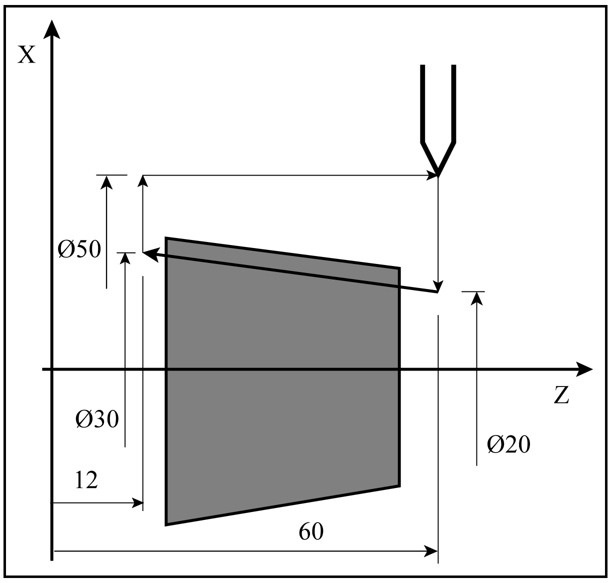


Fig. 4.4-3

☞ ***Notes*:**

– If in the thread-cutting block
both addresses F and E are also filled in,
F0 or E0 are specified
the control will indicate the error message *'2021 Thread programming error'*.

– In the course of execution of the command G33, the control considers the values of feed and spindle override 100% automatically, and the key Stop is ineffective.

– Because of following error of the servo system, run-in and run-out distances outside the material have to be provided for the tool at the forepart and the end of the thread in order to obtain constant lead all along the full length.

– Before issuing command on thread cutting, the state of calculation of constant cutting speed (G96) must be switched off.

4.5 Polar Coordinate Interpolation (G12.1, G13.1)

The polar coordinate interpolation is a control operation mode, in case of which the control goes along the path of the workpiece contour described in the orthogonal (Cartesian) coordinate system moving a ***linear axis*** and a ***rotary axis***, i.e. the path given by orthogonal coordinates is converted by the control into a path represented by polar coordinate data, moment by moment during the motion.

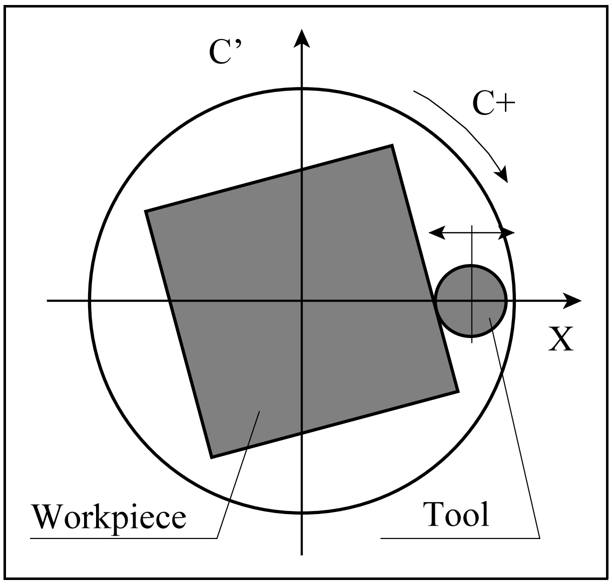


Fig. 4.5-1

The instruction Polar coordinate interpolation on

G12.1

switches the polar coordinate mode on. In the subsequent part of the program, the path of the milling tool ***in orthogonal coordinate system*** can be described by programming a linear axis and a ***rotary axis*** (a ***virtual linear*** axis). *The instruction must be issued in a separate block and no other instruction can be written beside*.

The instruction of Polar coordinate interpolation off

G13.1

switches the polar coordinate mode off. *The instruction must be issued in a separate block and no other instruction can be written beside*. After switching-on or reset, the control always enters into the state G13.1.

Selection of the Linear and the Rotary Axes

Before switching-on the polar coordinate interpolation, a linear axis and a rotary (virtual) axis have to be selected, these ones will be the axes participating in polar coordinate interpolation. Selection of axes is carried out using plane selection instructions G17, G18 and G19.

G17 X_P_ the address of a rotary (virtual) axis_
G18 Z_P_ the address of a rotary (virtual) axis_
G19 Y_P_ the address of a rotary (virtual) axis_

The first axis of the selected plane will always be ***the linear axis***. Parallel axis can be selected too.

The ***rotary axis*** specified in the plane selection instruction will be the rotary axis of the polar coordinate interpolation. Then, the instruction G12.1 will calculate with data given at the addresses of linear and rotary axes specified in the abovementioned way.

For example, the instruction

G17 X_ C_

designates the axis X for the linear one, and the axis C for the rotary one.

On the other hand, the instruction

G19 Y_ C_

designates the axis Y for the linear one, and the axis C for the rotary one.

Position of the Workpiece Zero Point in the Course of Polar Coordinate Interpolation

Before switching-on the polar coordinate interpolation, it is necessary to select such workpiece coordinate system, in which ***the center of rotation of the rotary axis coincides with the origin of the linear axis*** of the polar interpolation.
For example, if C is the rotary axis and X is the linear axis, the origin of the coordinate system will has to be chosen on the axis X in such a way that the X=0 position of the tool coincides with the axis of rotation of the circular axis (C).

If the center of rotation of the rotary axis does not coincide with the first axis of the selected plane (with the linear axis), in other words, if the center of rotation of the rotary axis does not coincide with the axis of rotation of the tool in the direction perpendicular to the linear axis, the machined workpiece will become deformed. The closer the tool moves to the origin the bigger the extent of distortion will be.
Since there is generally no axis in this direction, therefore this deviation cannot be compensated by zero point offset.
This deviation can be compensated in the parameter N0217 Polar Intp. Comp. Amount. The value of compensation must be written ***always at the address of the rotary axis*** in mm or in inch.

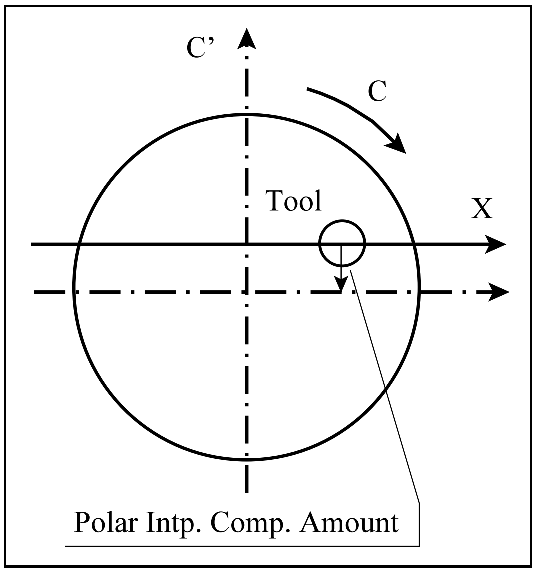


Fig. 4.5-2

Position of the Axes at the Moment of Switching the Polar Coordinate Interpolation on

Before switching the polar coordinate interpolation (the instruction G12.1), it is necessary to ensure ***being of the circular axis in the point of 0 position***. Position of the linear axis can be either ***negative*** or ***positive***, but ***it cannot be 0***.

Programming the Length Data in the Course of Polar Coordinate Interpolation

In the switched-on state of the polar coordinate interpolation, length coordinate data may be programmed on both axes belonging to the selected plane: the rotary axis in the selected plane functions as the second (virtual) axis. For example, if the axes X and C have been selected by means of instruction G17 X_ C_ , the address C can be programmed like the axis Y in the case of plane selection G17 X_ Y_.
In polar coordinate interpolation, the function G16 of programming in polar coordinate can also be applied. In this case, obviously, the polar radius is given at the first axis address of the selected plane, and the polar angle is given on the rotary axis.

Basically, it does not influence programming the virtual axis whether the first axis is programmed in diameter or not, the coordinate data on the virtual axis must always be given in radius. For example, if polar coordinate interpolation is executed in the plane XC, the value written at the address C must be given in radius independently of giving the address X in diameter or in radius.

Motion of the Axes Not Participating in Polar Coordinate Interpolation

The tool moves on these axes as it moves in normal case, independently of switched-on state of polar coordinate interpolation.

Programming the Circular Interpolation in the Course of Polar Coordinate Interpolation

When polar coordinate interpolation is switched on, a circle can be specified by radius in the way known already, or by programming the circle center coordinates. Having chosen the latter case, the addresses I, J and K have to be used in harmony with the selected plane, according to the following:

G17 X_P_ the address of a rotary (virtual) axis_ I_ J_
G18 Z_P_ the address of a rotary (virtual) axis_ I_ K_
G19 Y_P_ the address of a rotary (virtual) axis_ J_ K_

Use of Tool Radius Compensation in the case of Polar Coordinate Interpolation

The instructions G41, G42 can be used in the ordinary way when polar coordinate interpolation is switched on. Be careful to it that the tool position code in the rotary tool compensation group be Q=0. The following restrictions must be considered regarding its application:
Switch-on of polar coordinate interpolation (instruction G12.1) is possible in the state G40 only.
If G41 or G42 has been switched on in the state G12.1, G40 will have to be programmed before switching the polar coordinate interpolation (the instruction G13.1) off.

Programming Restrictions in the Course of Polar Coordinate Interpolation

The instructions listed below cannot be used when polar coordinate interpolation is switched on:
– plane change: G17, G18, G19;
– coordinate transformations: G52, G92;
– change of workpiece coordinate system: G54, ..., G59;
– positioning in the machine coordinate system: G53;
– G28 reference point return, G30 Pp: 2. 3. 4. return to the reference point;
– G31 measurement with deletion of remaining travel.

Feed in the Course of Polar Coordinate Interpolation

In the switched-on state of the polar coordinate interpolation, feed is interpreted, in the way ordinary in the case of the orthogonal interpolation, as tangential speed: it is the relative speed of the workpiece and the tool.
In the course of polar coordinate interpolation the control goes along a path specified in orthogonal coordinate system moving a linear axis and a rotary axis. As the tool center point approaches the axis of rotation of the circular coordinate, the rotary axis should take bigger and bigger steps in unit of time so as the tangential speed will be constant. On the other hand, the speed of the circular axis is limited by the maximum permissible speed of the rotary axis defined by parameter. For this reason, near the origin the control decreases the tangential feed step by step so that the speed of the rotary axis does not increase beyond all limits.
This figure illustrates the case of programming straight lines (1, 2, 3 and 4) parallel with the axis X. Δx displacement in

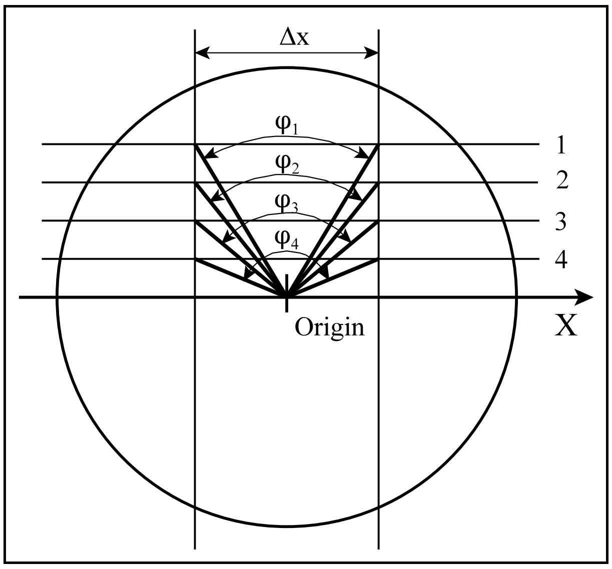


Fig. 4.5-3

unit of time belongs to the programmed feed. In the case of different straight lines (1, 2, 3 and 4), different angular displacements (φ_1, φ_2, φ_3, φ_4) belong to the Δx displacement. Apparently, the closer the machining gets to the origin, the larger angular displacement the rotary axis has to make in unit of time in order to keep the programmed feed.
If, in such cases, the angular speed exceeds the value set in the parameter N0305 Max Feed for the rotary axis, the control will decrease the tangential feed step by step.

An example:
This figure shows an example for use of polar coordinate interpolation. The axes participating in the interpolation are the following: axis X (linear axis) and axis C (rotary axis). The ***axis X*** is programmed ***in diameter***, while the ***axis C in radius***.

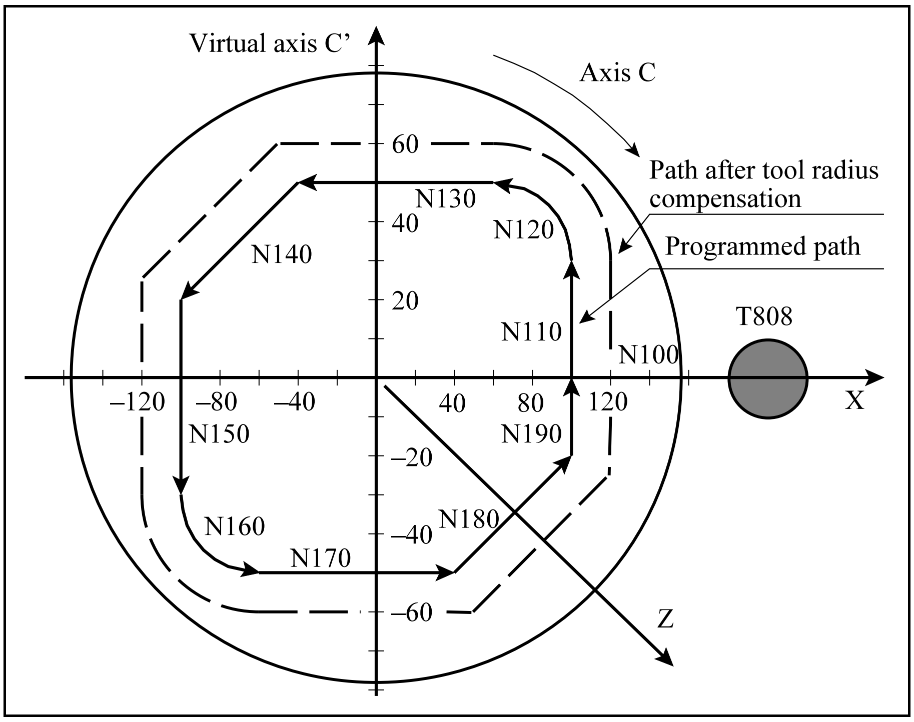


Fig. 4.5-4

```
...
N050 T808
N060 G59                       (The starting point of the
                               coordinate system G59 in the
                               direction X is the axis of
                               rotation of C)
N070 G17 G0 X200 C0            (Selection of the plane X, C;
                               positioning to the coordinate X≠0,
                               C=0)
N080 G94 Z-3 S1000 M3
N090 G12.1                (Polar coordinate interpolation On)
N100 G42 G1 X100 F1000
N110 C30
N120 G3 X60 C50 I-20 J0
N130 G1 X-40
N140 X-100 C20
N150 C-30
N160 G3 X-60 C-50 R20
N170 G1 X40
N180 X100 C-20
N190 C0
N200 G40 G0 X150
N210 G13.1                (Polar coordinate interpolation Off)
```

```
N220 G0 G18 Z100         (Tool  retraction, selection of the
                         plane X, Z)
...
```

4.6 Cylindrical Interpolation (G7.1)

If a cam should be milled on the mantle of a cylinder, cylindrical interpolation will be applied. In this case, the center line of the cylinder and the axis of rotation of a rotary axis must coincide.
The program is written by programming ***the linear axis parallel with the center line of the cylinder*** and ***the circular axis rotating the cylinder***. In the program, angular displacement of ***the rotary axis*** is given in ***degree*** which will be converted by the control, along the mantle, as a function of cylinder radius into the arc length measured in mm or inch. The displacement resulted from the interpolation will be reconverted by the control into angular displacement for the rotary axis.
The ***feed F*** specified in the course of cylindrical interpolation is always taken into account ***along the mantle of the cylinder***.
The instruction Cylindrical interpolation On **G7.1** Qr switches the cylindrical interpolation on, where

Q: the address of the rotary axis participating in the cylindrical interpolation;

r: the radius of the cylinder.

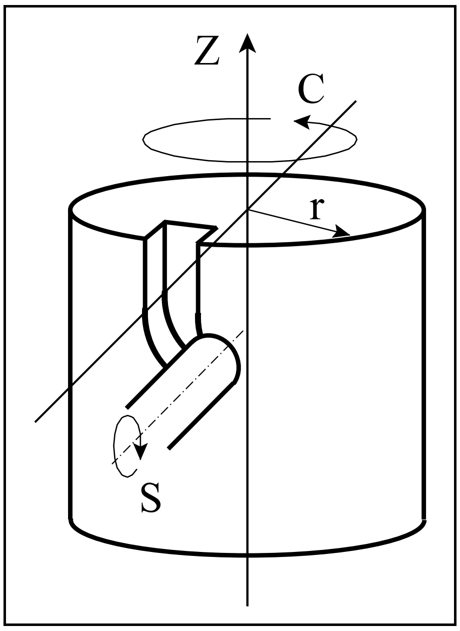


Fig. 4.6-1

For example, if the axis C is the rotary axis participating in the cylindrical interpolation and the cylinder radius is 50 mm, the cylindrical interpolation can be switched on by the instruction G7.1 C50.
In the subsequent part of the program, the path to be milled on the cylinder mantle can be described by specifying linear and circular interpolation. The coordinate for the linear axis must always be given in mm or in inch, while that of the rotary axis in degree (°).
The instruction Cylindrical interpolation Off **G7.1** Q0 switches the cylindrical interpolation off, i.e. the code G is the same as it was in the case of switching on, but at the address of the rotary axis 0 must be written.
The cylindrical interpolation switched on by the instruction G7.1 C50 in the abovementioned example, can be switched off by the command G7.1 C0.
After the turn-on, the end of the program or reset, the control will always arrive at the state of cylindrical interpolation Off.
The instruction G7.1 must be specified in a separate block.

Selection of the Linear and the Rotary Axes

Before switching-on the cylindrical interpolation, a linear axis and a rotary axis have to be selected, these ones will be the axes participating in cylindrical interpolation.
Selection of axes is carried out using plane selection instructions G17, G18 and G19.

G17 X_P_, or Y_P_ and the address of a rotary axis_

G18 X_P_, or Z_P_ and the address of a rotary axis_

G19 Y_P_, or Z_P_ and the address of a rotary axis_

The cylindrical interpolation interprets G2 and G3 circular interpolation directions and the direction of tool radius compensation (G41 and G42) on the basis of the plane selected and the

linear axis belonging to the plane. The circular axis will be the other axis of the plane. A parallel axis can also be selected.
For example, let the axis Z be parallel with the center line of the cylinder, and the axis C be the axis of rotation of the cylinder. In this case, the linear axis and the rotary axis can be assigned using either the instruction

```
G18 Z_ C_
```

or the instruction

```
G19 Z_ C_
```

before switching the cylindrical interpolation on.

Circular Interpolation

It is possible to define circular interpolation in cylindrical interpolation mode, however by specifying the radius R only.
In the case of cylindrical interpolation, circular interpolation is not possible by giving the circle center point (I, J, K).
The circle radius is always interpreted in mm or in inch, never in degree.
For example, circular interpolation between axes Z and C can be specified in two ways:

```
G18 Z_ C_                               G19 C_ Z_
G2 Z_ C_ R_                             G3 C_ Z_ R_
G3 Z_ C_ R_                             G2 C_ Z_ R_
```

If the intension is to define the same path by G18 and G19 plane selection, the circle directions will interchange compared to each other.

Application of Tool Radius Compensation in the case of Cylindrical Interpolation

The instructions G41, G42 can be used in the usual manner in the switched-on state of cylindrical interpolation. The following restrictions are in effect regarding its application:
– Switching the cylindrical interpolation on (instruction G7.1 Qr) is possible in the state G40 only.
– If G41 or G42 has been switched on in the cylindrical interpolation mode, G40 must be programmed before switching the cylindrical interpolation off (instruction G7.1 Q0).

Programming Restrictions in the Course of Cylindrical Interpolation

The following instructions are not available in the switched-on state of cylindrical interpolation:
– plane selection: G17, G18, G19;
– coordinate transformations: G52, G92;
– change of workpiece coordinate system: G54, ..., G59;
– positioning in the machine coordinate system: G53;
– circular interpolation by specifying the circle center point (I, J, K);
– drilling cycles.

An example:
This figure illustrates a path to be milled 3 mm deep on the mantle of a cylinder with radius of R=28.65 mm. The milling tool T is parallel with the axis X.

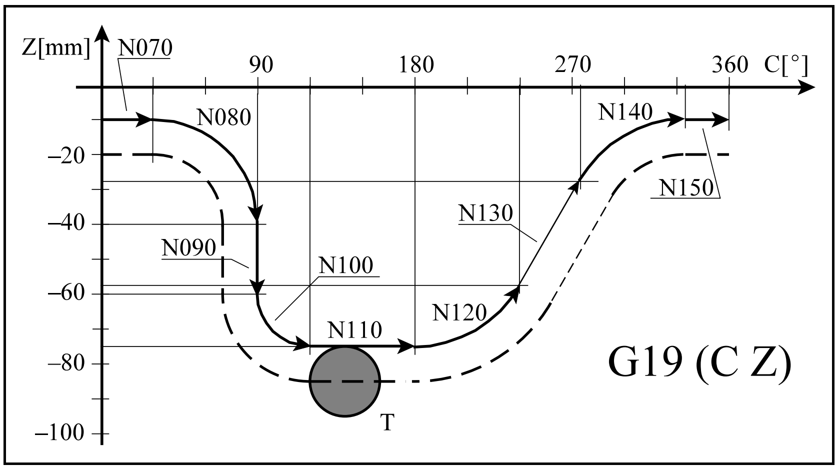


Fig. 4.6-2

On the mantle of the cylinder, the displacement corresponding to the 1 degree (1°) is:

$$28.65mm \cdot \frac{1^\circ}{180^\circ} \cdot \pi = 0.5mm$$

Arrangement of the axes shown in the figure corresponds to the plane selection G19.

```
(CYLINDRICAL INTERPOLATION)
...
N030 G19 Z-20 C0              (G19: selection of the plane C-Z)
N040 G1 X51.3 F100            (3 mm depth of cut)
N050 G7.1 C28.65              (Switching the cylindrical
                              interpolation On, rotary axis: C,
                              cylinder radius: 28.65 mm)
N060 G1 G42 Z-10 F250
N070 C30
N080 G2 Z-40 C90 R30
N090 G1 Z-60
N100 G3 Z-75 C120 R15
N110 G1 C180
N120 G3 Z-57.5 C240 R35
N130 G1 Z-27.5 C275
N140 G2 Z-10 C335 R35
N150 G1 C360
N160 G40 Z-20
N170 G7.1 C0                  (Switching the cylindrical
                              interpolation Off)
N180 G0 X100
...
```

5 Coordinate Data

5.1 Absolute and Incremental Programming (G90, G91)

The input coordinate data can be specified as absolute and incremental values too. In the case of absolute data specification, the coordinates of the end point must be given to the control, while in the case of incremental data it is the distance to be travelled in the block that must be programmed.

G90: Programming the absolute data specification

G91: Programming the incremental data specification

The G90 and G91 are modal functions. At the time of power-on, the control, according to the bit #7 G91 of the parameter N1300 DefaultG1, decides which of the two states will apply. At the end of the program or in case of reset, the code set in this parameter is effective too.
Motion to an absolute position is feasible after a reference point return only.

An example:
On the basis of this figure, the motion can be programmed in the following two ways:

```
G90 G01 X100 Z20
G91 G01 X60 Z-40
```

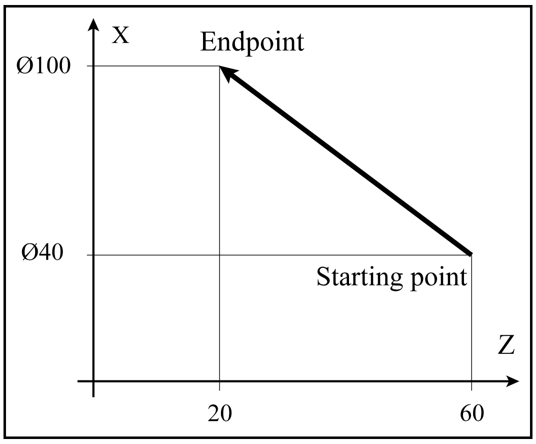


Fig. 5.1-1

The ***operator I*** is effective in the state G90 of absolute data specification. ***It applies to the coordinate only after the address of which it stands***. Its meaning is: incremental data.
Another way to solve the example above is:

```
G90 G01 XI60 ZI-40
G01 XI60 Z20
G01 X100 ZI-40
```

In the case of using ***multicharacter axis name***, if the name ***ends with number***, e.g. axis X2, the operator I will have to be written after the sign=:

```
X2=I100
```

If the ***addresses U, V and W*** are not assigned for axes, they can be used to indicate incremental motions in the directions X, Y and Z:

	Address of absolute command	Address of incremental command
Motion command in the direction X	X	U
Motion command in the direction Y	Y	V
Motion command in the direction Z	Z	W

Taking the abovementioned into account, the solution of the example is:

```
      G90 G01 U60 W-40
      G01 U60 Z20
      G01 X100 W-40
```

5.2 Inch/Metric Conversion (G20, G21)

By programming appropriate code G, the input data can be specified either in inch unit system or in metric unit system.

G20: Choosing the inch unit system

G21: Choosing the metric unit system

The unit system chosen will be in effect until a command of contrary meaning is issued, therefore the codes G20 and G21are modal ones.

At the time of power-on, the control, according to the bit #3 G20 of the parameter N1300 DefaultG1, decides which of the two states will apply. At the end of the program or in case of reset, the code set in this parameter is effective too.

For example:

```
      G21 G0 G54 X200 Z50 (positioning to X=200 mm, Z=50 mm)
      G20 X2 Z1 (positioning to X=2 inch, Z=1 inch)
```

Changing the unit system influences the following items:

– Coordinate and compensation data, (mm/inch);

– Feed rate (mm/min, inch/min, mm/ford, inch/ford);

– Constant cutting speed (m/min, feet/min);

– The values of position, compensation, zero point offset and feed rate are always displayed in the unit system that has been chosen.

– When macro variables (offset, position data etc.) are being read, data will be read out in the unit system chosen.

– Incremental jog and handwheel motion step;

– Feed rate of the manual move (jog).

5.3 Programming in Diameter or Radius

Since the cross-section of the workpiece machined on lathes is usually a circle and the workpiece can be controlled by diameter measuring, it is practical to specify the dimensions in direction of the axis X in diameter. If there is axis Y on the machine, it will be generally practical to manage the dimension in direction of the axis Y in diameter too. For each axis, it can be specified in the bit #0 DIA of the parameter N0106 Axis Properties whether the control interprets the dimension in diameter or in radius:

in the case of **programming in radius**:

#0 DIA=0

in the case of **programming in diameter**:

#0 DIA=1

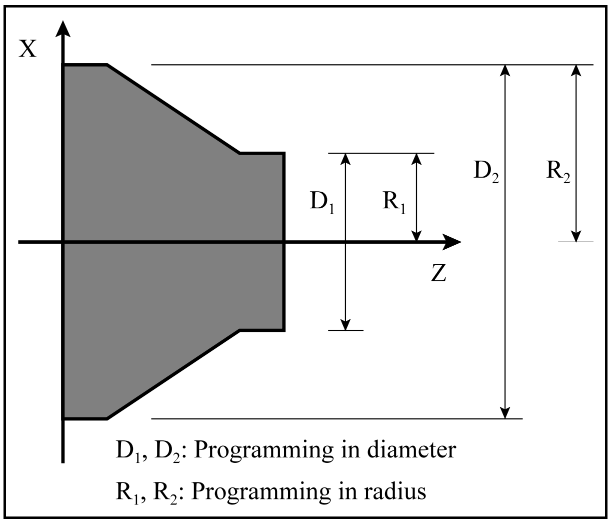

X
Z
D_1
R_1
D_2
R_2
D_1, D_2: Programming in diameter
R_1, R_2: Programming in radius

Fig. 5.3-1

If the parameter is set for programming in diameter, the following will have to be taken into account:

Case	Note
Absolute motion command	Specified with diameter value
Incremental motion command	Specified with diameter value (on the basis of the figure $D_2 - D_1$)
Coordinate offset and zero point offset	Specified with diameter value
Tool length compensation	Specified with diameter value
In canned cycles, parameters concerning axes managed in diameter, such as cutting depth	Always specified with radius value
The value of R and I, J and K in case of specifying circular interpolation	Always specified with radius value
Displaying the axis position	As diameter value
Feed rate in direction of the axis managed in diameter	Always in radius/rev or radius/min
Step value in jog or handwheel mode	Step of 1μm will be taken in diameter if one increment is chosen

5.3.1 Switching between in Radius and in Diameter Programming (G10.9)

It is specified by the value of

the bit #0 DIA of the parameter N0106 Axis Properties

whether the programming on a given axis will be done in radius or in diameter. If the value of the bit

=0: the axis will be programmed in radius;

=1: the axis will be programmed in diameter.

The mode of programming can be switched anywhere in the part program. It is specified by the value of

the bit #5 MGD of the parameter N0106 Axis Properties

whether the switching will be executed from PLC or by code G. If the value of the bit

=0: the switching will be executed through PLC signal;

=1: the switching will be executed by the code G10.9.

The manner of the switching from PLC is defined by the machine builder.

Hereunder, the switching by the code G10.9 will be described.

The instruction

G10.9 v

switches between programming in radius and programming in diameter, where

v: address of arbitrary linear axes, e.g. X, Y ... etc.

If the value written at the axis address:

=0: the axis will be programmed in radius;

=1: the axis will be programmed in diameter.

For example, in basic case, the axes X and Y on a milling machine are programmed in radius. If, on a given section of the part program, it is practical to specify the value of X and Y in diameter, the following will have to be written in the part program:

```
    ... (X, Y are specified in radius)
    ...
    G10.9 X1 Y1 (programming the diameter)
    ... (X, Y are specified in diameter)
    ...
    G10.9 X0 Y0(programming the radius)
    ... (X, Y are specified in radius)
    ...
```

If, on a lathe, the axis is programmed in diameter, but in the case of milling to be performed using polar interpolation and axes X and C, the intension is to program the X in radius, then:

```
    ... (X is specified in diameter, turning)
    ...
    G10.9 X0 (programming the radius)
    G12.1 (polar interpolation On)
    ... (X, is specified in radius, milling)
    ...
    G13.1 (polar interpolation Off)
    G10.9 X1 programming the diameter)
    ... (X is specified in diameter, turning)
    ...
```

The code G10.9 must be given always in a separate block!

When the program ends or reset is done, the programming mode set by the code G10.9 will be deleted, and the control returns to the basic state set in the bit #0 DIA of the parameter N0106 Axis Properties.

5.4 Data Specification in Polar Coordinates (G15, G16)

The values of the programmed coordinates can also be input as radius given in mm or in inch, and angle given in degree.

G16: Data specification in polar coordinates on

G15: Data specification in polar coordinates off

G15 and G16 are modal functions. After power-on, at the end of the program or in case of reset, the control turns to the state G15.

Data specified in polar coordinates are interpreted in the plane defined by the codes G17, G18 and G19. In the course of data specification, the control considers the address of the first axis of the plane to be the radius, and the second axis to be the angle.

G17 X_P (radius) Y_P (angle)

G18 Z_P (radius) X_P (angle)

G19 Y_P (radius) Z_P (angle)

X_P, Y_P, Z_P could be the basic axes X, Y, Z or axes parallel with them.

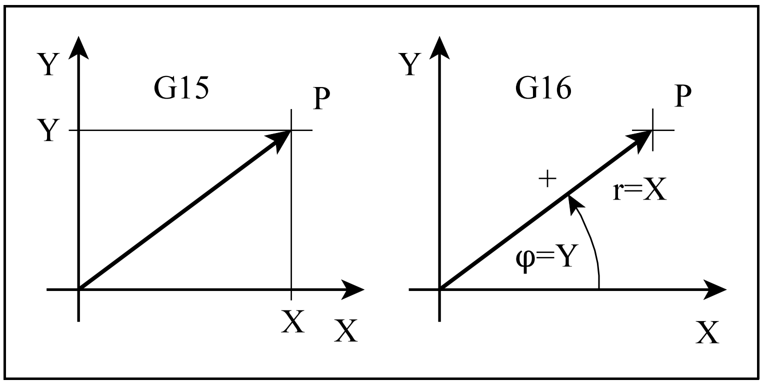


Fig. 5.4-1

In the case of angle specification, ***counter-clockwise*** is the ***positive*** direction of the angle and ***clockwise*** is the ***negative*** direction of the angle.
The control considers data of other axes being outside of the selected plane to be data specified in Cartesian coordinates.

The radius and the angle can be specified as absolute data and incremental data as well.

When ***the radius is specified as absolute data***, the origin of the actual coordinate system will be the origin of the polar coordinate system:

```
G90 G16 X100 Y60
```
Both the angle and the radius are absolute data, the tool moves to the point of 100 mm radius and of 60° angle.

```
G90 G16 X100 YI40
```
The angle is an incremental data. The tool moves to the point of 100 mm radius being further on by 40° compared to the previous angle position.

When ***the radius is specified as incremental data***, the control moves the given radius from the axis positions specified at the beginning of the block in the direction of the angle given:

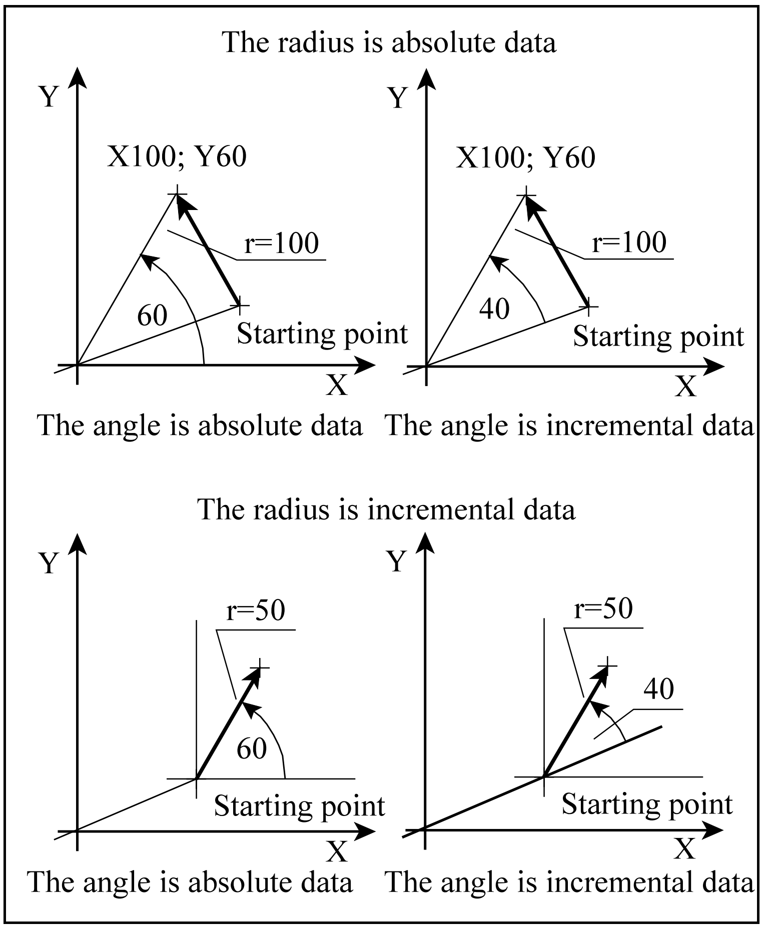

The radius is absolute data
Y
X100; Y60
r=100
60
Starting point
X
The angle is absolute data
Y
X100; Y60
r=100
40
Starting point
X
The angle is incremental data
The radius is incremental data
Y
r=50
60
Starting point
X
The angle is absolute data
Y
r=50
40
Starting point
X
The angle is incremental data

Fig. 5.4-2

```
G90 G16 XI50 Y60
```
The control measures the radius of 50 mm from the starting point and moves to the absolute angle of 60°.

```
G91 G16 X50 Y40
```
The control measures the radius of 50 mm from the starting point and moves to the angle of 40° measured from the starting angle.

Programming a circle is also possible when the data specification in polar coordinates G16 is on. The circle can be specified both by radius and by I, J and K. But in the latter case, the addresses ***I, J and K*** will ***always*** be considered by the control to be ***Cartesian data***.
When the origin of the actual coordinate system coincides with the center of the circle, multi-revolution circle or spiral can also be programmed using data specification in polar coordinates:

```
G17 G16 G90 G02 X100 Y-990 Z50 R-100
```

In the block above, a clockwise spiral of 2 3/4 revolutions is specified. When a multi-revolution circle is to be programmed, attention has to be paid on programming negative polar angle in case of direction G2, and on programming positive polar angle in case of direction G3.
The addresses in the following instructions are not considered by the control to be polar coordinate data even in the case when the state G16 is switched on:
– G10 coordinates in setup instruction;
– G52 coordinate offset;
– G92 coordinate setup;
– G28, G30 coordinates of an intermediate point;
– G53 positioning in the machine coordinate system;
– G68 the central point of rotation of the coordinate system;
– G51 the central point of scaling;
– G50.1 the central point of reflection.

An example: Milling a hexagon

```
    N1 G90 G17 G0 X120 Y120 F120
    N2 G16 G1 X100 Y60
    N3 Y120
    N4 Y180
    N5 Y240
    N6 Y300
    N7 Y0
    N8 Y60
    N9 G15 G0 X120 Y120
```

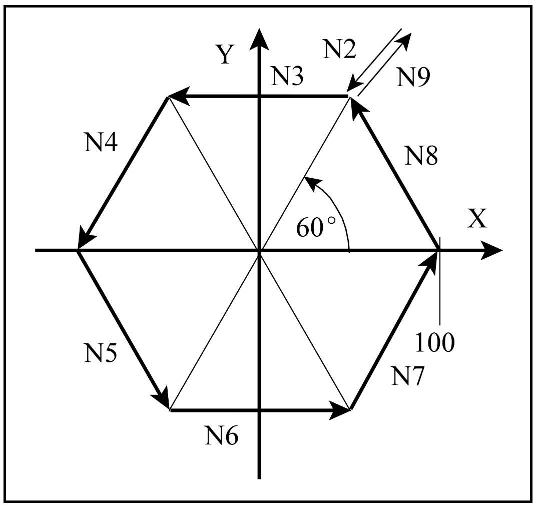


Fig. 5.4-3

5.5 Specification and Accuracy of Coordinate Data

The control interprets the decimal point according to the unit system applied:
– X2.134 means 2.134 mm or 2.134 inch;
– B24.236 means 24.236 degrees, when angle data is specified at the address B.
Use of the decimal point is not mandatory:
– X325 means 325 mm, for example.
The leading zeros can be eliminated:
– .032=0.032
The following zeros after the decimal point can be eliminated:
– 0.320=.32
Coordinate data can be specified with accuracy of up to 15 decimal digit.

5.6 Managing the Rotary Axis Roll-Over

This function can be used in case of axes that rotate, i.e. when an axis address (for eample C) is assigned to a rotary axis.
Managing the roll-over means that the position on the given axis is registered by the control not in between plus and minus infinities, but in between 0° and 360° for example, taking the axis periodicity into account.

Designation of an Axis to be Rotary Axis and the Effect Produced

This designation has to be done by bit setting #1 ROT=1 in the parameter N0106 Axis Properties . For rotary axis,
– the control ***does not do the inch/metric conversion***;
– **the roll-over management can be enabled**.

Enabling the Roll-Over Management

This function is activated by bit setting #0 REN=1 in the parameter N0107 RollOver Control . If the value of the bit REN:
=0: the control will manage the rotary axis as it manages the linear axes, and filling the further parameters will not produce any effect;
=1: the control will apply the roll-over management for the rotary axis, and the essence of this is as follows.

Specification of the Distance Travelled per Revolution

The distance travelled per one revolution of the axis can be specified in the parameter N0108 RollAmount. Thus, if the axis rotates 360° per one revolution, the value to be written in the appropriate parameter will have to be 360. The system can manage arbitrary periodicity not only that of 360°.
Using the abovementioned parameter settings, the control displays the position of the rotary axis always in the range of 0°- +359.999° independently of the direction of rotation and the number of revolutions the rotary axis performed.

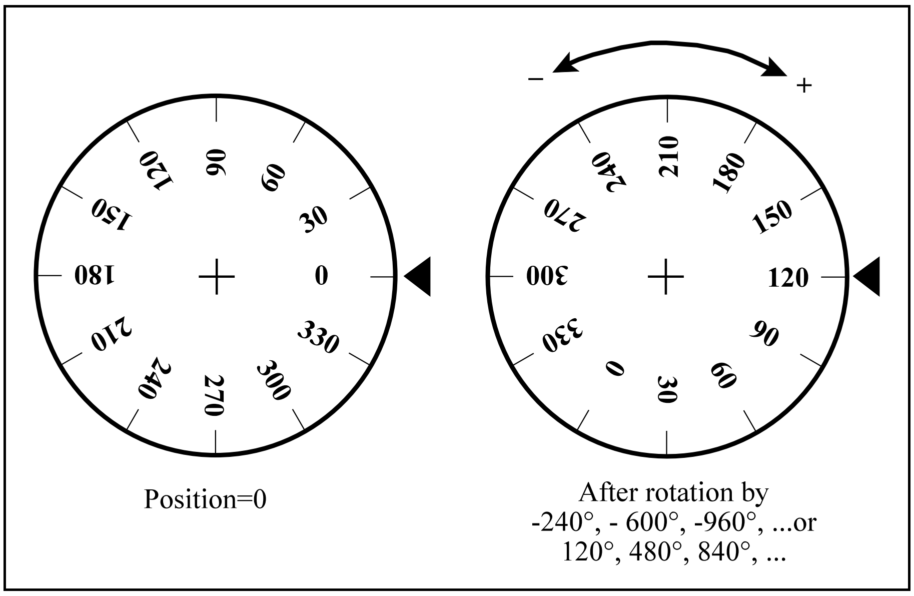


Fig. 5.6-1

In the setting opportunities listed below, the bit state #0 REN=1 of the parameter N0107 RollOver Control and setting of the parameter N0108 RollAmount is supposed.

Rotation to the Specified Position: the Basic Case of Rotation

If the settings of the parameter N0107 RollOver Control are

#2 ABS=0 and #1 ASH=0,

in the case of absolute programming (G90) the rotation will be performed to the specified position. The value of displacement will always be less than the value specified in the parameter RollAmount, i.e. during motion the control always cuts the full revolutions.

An example:

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C210
G90 C570
G90 C930
. . .
```

it will always rotate 210 degree (ΔC=210°) in the positive direction, having cut the full revolutions, and the end position will be 210°.

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C-210
G90 C-570
G90 C-930
. . .
```

it will always rotate 210 degree (ΔC= –210°) in the negative direction, having cut the full revolutions, and the end position will be 150°(= –210°).

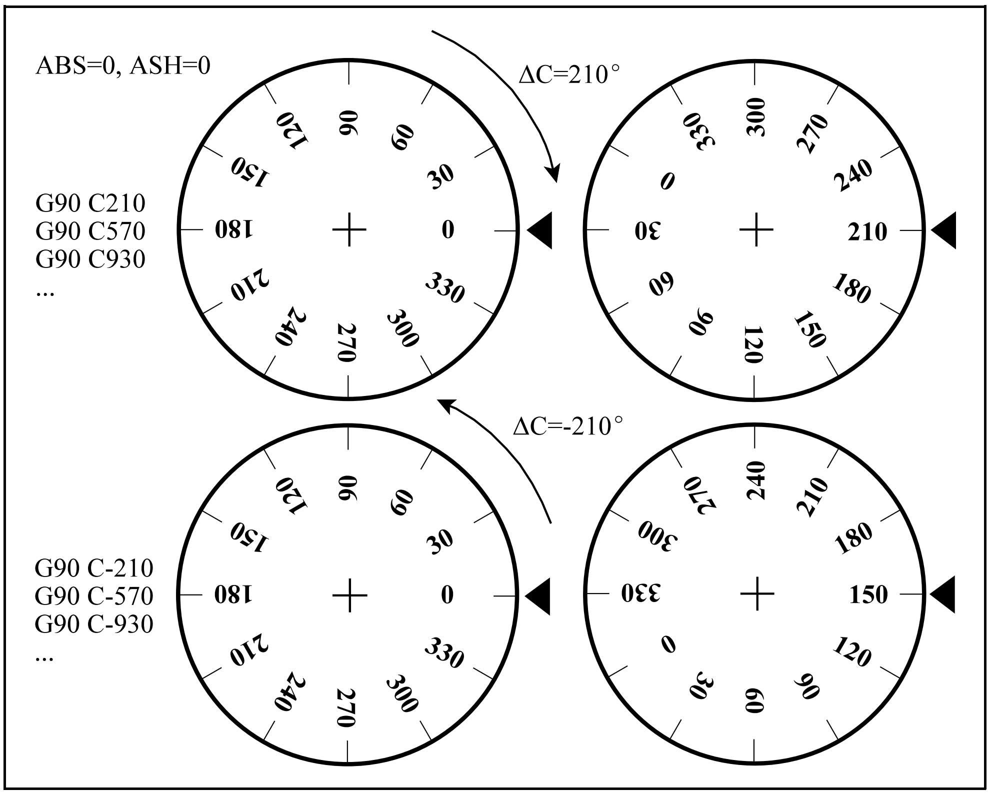


Fig. 5.6-2

Rotation to the Specified Position in the Shorter Way

If the settings of the parameter N0107 RollOver Control are

#2 ABS=0 and #1 ASH=1,

in the case of absolute programming (G90) the rotation will be performed to the specified position and in the direction belonging to the shorter way. The value of displacement will always be less than the value specified in the parameter RollAmount, i.e. during motion the control always cuts the full revolutions.

An example:

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C210
G90 C570
G90 C930
...
```

it will always rotate 150 degree (ΔC=－150°) in the negative direction (in the shorter way), having cut the full revolutions, and the end position will be 210°.

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C-210
G90 C-570
G90 C-930
...
```

it will always rotate 150 degree (ΔC=150°) in the positive direction (in the shorter way), having cut the full revolutions, and the end position will be 150°(= -210°).

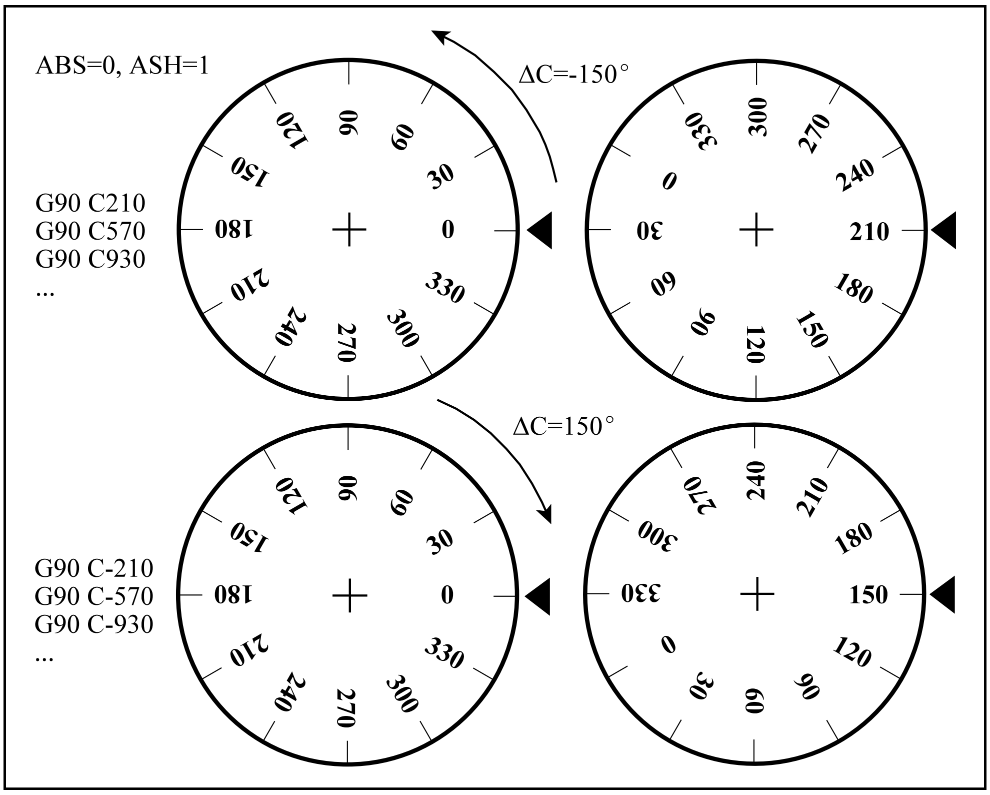


Fig. 5.6-3

Rotation to the Absolute Value of the Specified Position in the Direction Defined by the Sign

If the settings of the parameter N0107 RollOver Control are

#2 ABS=1 and #1 ASH=0,

in the case of absolute programming (G90) the rotation will be performed to the absolute value of the specified position and in the direction defined by the sign. The value of displacement will always be less than the value specified in the parameter RollAmount, i.e. during motion the control always cuts the full revolutions.

An example:

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C30
G90 C390
G90 C750
...
```

it will always rotate 30 degree (ΔC=30°) in the positive direction, having cut the full revolutions, and the end position will be 30°.

Let the starting position of the axis C be

C=0

In accordance with the instructions

```
G90 C-30
G90 C-390
G90 C-750
...
```

it will always rotate 330 degree (ΔC= −330°) in the negative direction, having cut the full revolutions, and the end position will be 30°.

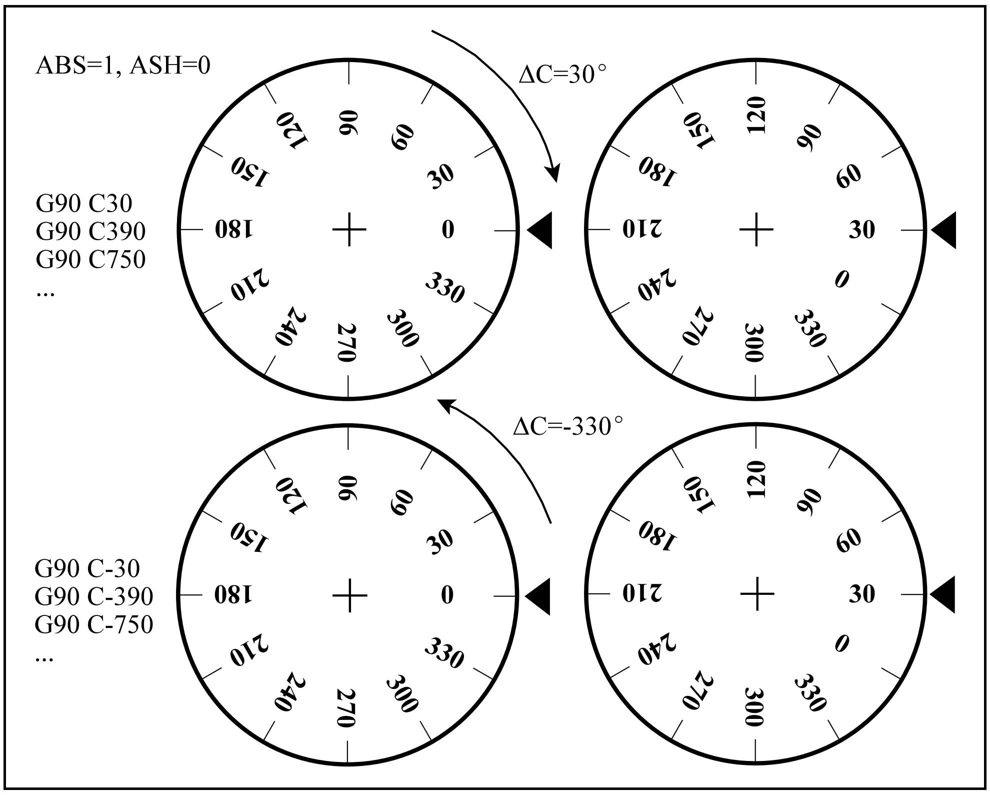


Fig. 5.6-4

☞ ***Note:*** The 0 (zero) is a positive number! Therefore, if position 0 is programmed, the motion will be performed in the positive direction.

Let the starting position of the axis C be

C=0

In accordance with the instruction

```
G90 C0
```

it will rotate 330 degree (ΔC=330°) in the positive direction, and the end position will be 0°.

If the motion from 30° to 0° in the negative direction is required, the block

```
G90 C-360
```

will has to be programmed. In this case, it will rotate 30 degree (ΔC= −30°) in the negative direction, and the end position will be 0°.

Motion of a Rotary Axis in the Case of Incremental Programming

In the case of programming incremental data specification, the motion will be performed in the direction defined by the programmed sign.
If the setting of the parameter N0107 RollOver Control is
REN=0,
the control does not apply the parameter Roll Amount for the travel programmed incrementally, so multi-rotation displacement can be programmed using incremental specification.
Let the starting position of the axis C be
C=0
During execution of the blocks

```
G91 C30 (displacement 30 degree)
G91 C390 (displacement 390 degree)
G91 C750 (displacement 750 degree)
...
```

the control does not cut the full revolutions. On the other hand, the position of the axis C on the position display will always be 30 degree at the end of the motion, because the value set in the parameter RollAmount will always be valid for displaying the position.
If the setting of the parameter N0107 RollOver Control is
REN=1,
the control applies the parameter Roll Amount for the travel programmed incrementally too, so displacement greater than one rotation cannot be programmed even if using incremental specification.
Let the starting position of the axis C be
C=0
During execution of the blocks

```
G91 C30 (displacement 30 degree)
G91 C390 (displacement 30 degree)
G91 C750 (displacement 30 degree)
...
```

the control cuts the full revolutions. On the other hand, the position of the axis C on the position display will always be 30 degree.

6 Feed

6.1 Rapid Traverse

Positioning is executed by rapid traverse activated by the command G0. Apart from the G0 positioning, the positioning phases of the commands G53, G28,G30 and the cycles are executed by rapid traverse too. The value of the positioning rapid traverse for each axis ***is set in parameter by the builder of the machine, in mm/min, inch/min or degree/min***. The rapid traverse values for the axes can differ from each other.
When several axes perform rapid traverse simultaneously, the value of the resultant feed will be calculated by the control in such a way so that neither of the speed components projected on the axes will exceed the rapid traverse value given in parameter and valid for that given axis, and the positioning will be performed in a minimum period of time.
The value of the rapid traverse can be modified by the rapid traverse percent (override) switch. Operation of the rapid traverse override is defined in the PLC program by the builder of the machine. The description of operation is contained in the manual provided by the builder of the machine.
The value of the rapid traverse override does not exceed 100%.
The rapid traverse will always be stopped by the position 0% of the rapid traverse percent switch
.
If there is no valid reference point, the decreased rapid traverse values defined by the builder of the machine will be valid for the axes in the parameter field until reference point is done.
In the case of moving the slide by the slide-actuating (jog) buttons, the rapid traverse differs from the positioning rapid traverse, and it has a value different for each axis and set in parameter too. Evidently, its value is less than that of the positioning rate in order that the human response time can be calculated in for stopping.

6.2 Cutting Feed

Feed is programmed
at the address **F**.
The programmed feed will be valid in blocks of linear interpolation (G01) and blocks of circular interpolation (G02 and G03).
The unit of the feed is determined by ***the codes G94 and G95***.
The feed is calculated by the control tangentially along the programmed path.

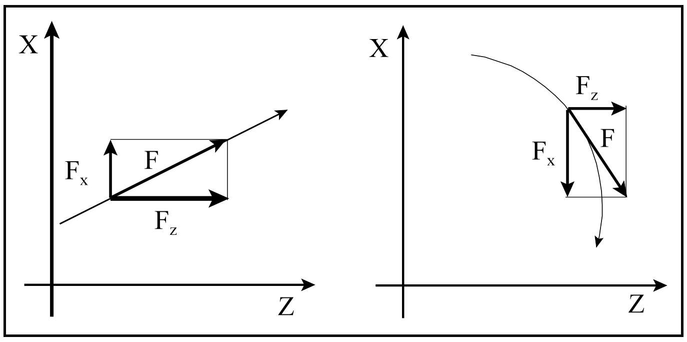


Fig. 6.2-1

F : tangential feed (programmed value)
F_x: feed component in the direction X
F_z: feed component in the direction Z

$$F = \sqrt{F_X^2 + F_Z^2}$$

The programmed feed can be modified using the feed percent (override) switch, excluding the G63, the percent switch and the inhibition of the stop states.

Operation of the feed override is defined in the PLC program by the builder of the machine. The description of its operation and value limit are contained in the manual provided by the builder of the machine.
The feed is a modal value.
After the power-on, the control will have the value specified in parameter. Namely, the initial value of the feed will be received from the parameter N0300 Default F G94 in the state G94, and from the parameter N0301 Default F G95 in the state G95.
For a given machine, the maximum programmable feed is limited for each axis by the builder of the machine in the parameter field. The value set here is always interpreted in minute dimension. When the switch DRY RUN is in switched-on position, this value, at the same time, is the speed of the feed motions.
The value of feed can be given with an accuracy of up to 15 decimal digit. Decimal point can be used. It is always a positive number.

6.2.1 Feed per Minute (G94) and Feed per Revolution (G95)

In the program, the unit of feed can be specified by the codes G94 and G95:

G94: feed per minute
G95: feed per revolution

The term 'feed per minute' refers to the feed specified in the units of mm/min, inch/min or deg /min.
The term 'feed per revolution' refers to the feed accomplished in a revolution of the spindle and specified in the units of mm/rev, inch/rev or deg/rev.
They are modal values.
After power-on, reset or program end, the bit #0 G95 of the parameter N1301 DefaultG2 determines whether the control will get into the state G94 or G95.
The state G94/G95 does not affect the rapid traverse which has to always be interpreted in 'per minute' dimension.

6.3 Feed Control Functions

The feed control functions are necessary
– in the case of ***machining corners***,
– in such cases when it is required by the technology so that the ***switches override and stop*** must be ***inhibited***.

In the case of machining corners in the mode of continuous cutting, the slides, because of their inertia, are not able to follow the path commands issued by the control. Depending on the feed, the tool will round the corner more or less.

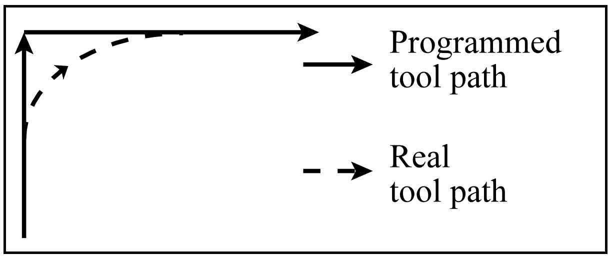


Fig. 6.3-1

If sharp corners are required on the workpiece, the control must be told to decelerate at the end of the motion, to wait until the axes stop, and to start the subsequent motion following this only.

6.3.1 Exact Stop at the End of the Block (G9)

This function is ***not a modal one***, it is valid in that block only in which it was programmed.
At the end of the bloc in which it was specified, the control decelerates after execution of interpolation, ***it stops and waits for the in-position signal of the measuring system***.
If the in-position signal does not arrive even after expiration of the time set in parameter, the control will send the error message '2501 Position error'.
This function is used for machining sharp corners.

6.3.2 Exact Stop Mode (G61)

This is a modal function. It can be cancelled by the commands G62, G63 and G64.
The control decelerates ***at the end of each block***, ***it stops and waits for the in-position signal of the measuring system***, and starts the subsequent interpolation cycle following this only.
If the in-position signal does not arrive even after expiration of the time set in parameter, the control will send the error message '2501 Position'.
This function is used for machining sharp corners.

6.3.3 Continuous Cutting Mode (G64)

This function is a modal one. ***This is the state the control gets into after power-on, reset or program end***. It can be cancelled by the codes G61, G62 and G63.
In this mode of operation, the motion does not stop after execution of the interpolation, the axes do not wait for the in-position signal of the measuring system, but the interpolation of the next block starts immediately.
In this mode of operation, sharp corners cannot be machined because the control rounds them at transitions.

6.3.4 Override and Stop Inhibition Mode (G63)

This function is a modal one. It can be cancelled by the codes G61, G62 and G64.
In this mode of operation, ***the percent switches of feed and spindle (override) as well as stopping the feed are ineffective***. The control interprets the percent values as 100% independently of the positions of the switches. It does not decelerate after execution of the interpolation, but it starts the next interpolation cycle immediately.
This mode of operation can be used for machining threads.

6.3.5 Automatic Feed Override at Inner Corners (G62)

This is a modal function. It can be cancelled by the commands G61, G63 and G64.
In the case of machining inner corners, the force acting on the tool increases on the sections before and after the corner. In order that the tool will not flutter and the surface will remain appropriate, the control reduces the feed automatically on the sections before and after the inner corners, when the G62 is switched on.
The corner override will be effective under the following conditions:

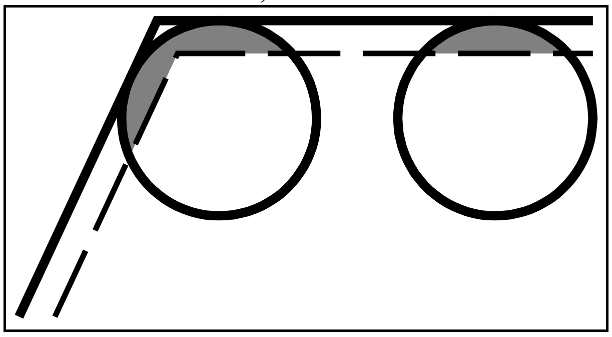


Fig. 6.3.5-1

– if the planar tool radius compensation is switched on (G41, G42);
– between the blocks G1, G2 and G3;
– in the case of motions in the selected plane;
– if the tool machines the corner inside;
– if the angle of the corner less than an angle specified in parameter;
– before and after the corner, at a distance specified in parameter.
The feed override function acts for all the following four possible transitions: linear - linear; linear - circular, circular - line, circular- circular.

The value of the internal angle φ can be set in the parameter N1409 CornAngle, in the angle range of 1° - 180°. Deceleration and acceleration will be commenced and finished at the distance L_d before and at the distance L_a after the corner, respectively. In cases of circle arcs, the distances L_d and L_a will be taken into account by the control along the arc. The distances L_d and L_a can be specified in the parameters N1407 DecDist and N1408 AccDist, respectively.

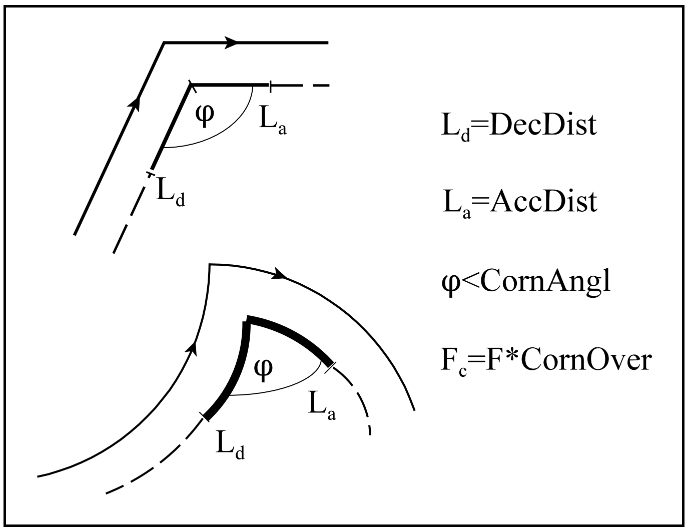


Fig. 6.3.5-2

Writing the ratio between 0 and 1 in the parameter N1410 CornOver there can be specified the value up to which the control has to reduce the feed at inner corners. The feed will be

F*CornOver

where F is the programmed feed. Even the override switch produces effect on the feed obtained, too.
If the aim is to program exact stop in the state G62, G09 will have to be written in the given block.

6.4 Automatic Feed Override in the Cases of Inner Circle Arcs

When the planar tool radius compensation is switched on (G41, G42), the programmed feed during circular interpolation will be effective along the tool center. In the case of machining inside circle arcs, the control reduces the value of the feed automatically so that the programmed feed will be effective along the cutting radius. The value of the feed in the center of the tool radius is:

$$F_C = \frac{R_C}{R}F$$

where F_c: the feed of the center of the tool radius (corrected feed);
R: the programmed circle radius;
R_c: the corrected circle radius;
F: the programmed feed.

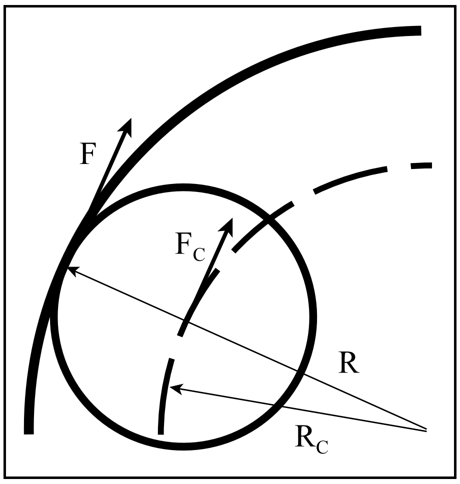


Fig. 6.4-1

The lower limit of automatic feed reduction is determined by the parameter N1406 CircOver, writing in which a ratio between 0 and 1 there can be specified the minimum value of feed reduction, i.e. the condition

$$F_c \geq F*CircOver$$

will be satisfied. The override for circle radius will be multiplied by the feed override and the corner override, and then it will be issued.

7 Acceleration

Acceleration describes how velocity changes over the course of time. The shorter the time necessary to reach a given velocity is, the great acceleration will be.
The higher acceleration we would like to reach, the higher power for motors and drives is required.
During motion, the value of forces acting on the machine, after all the load on the machine is in direct proportion to the acceleration come into being.
Parameters of acceleration for each axis are set by the builder of the machine on the basis of the two abovementioned considerations

The control accelerates always the value of the tangential (vectorial) feed. It calculates the value of acceleration in such a way so that the value of the axis component of acceleration does not exceed, on none of the axes, the value set for the given axis.
Acceleration of two kinds can be set:
– linear acceleration, and
– bell-shaped acceleration.

In the case of ***linear acceleration***, the value of acceleration is constant during acceleration/deceleration, the control increases the feed at start or reduces the feed at stop in accordance with a linear function.
The value of acceleration can be set in parameter for each axis.

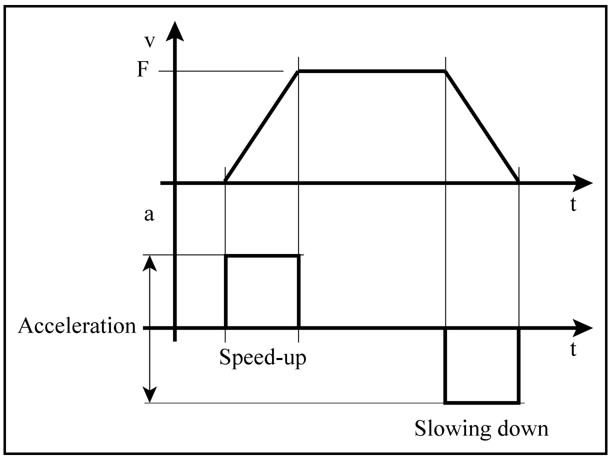


Fig. 7-1

In the case of ***bell-shaped acceleration***, the value of acceleration changes during acceleration/deceleration, it increases linearly until it reaches the set value of acceleration, or it decreases linearly until it reaches the target speed. As a consequence of this, the shape of the feed leading and trailing branches is a bell-shaped curve as a function of time, and because of this the acceleration of this kind is called 'bell-shaped'.
The time T, during which the control reaches the set value of acceleration, can be set in parameter for each axis.

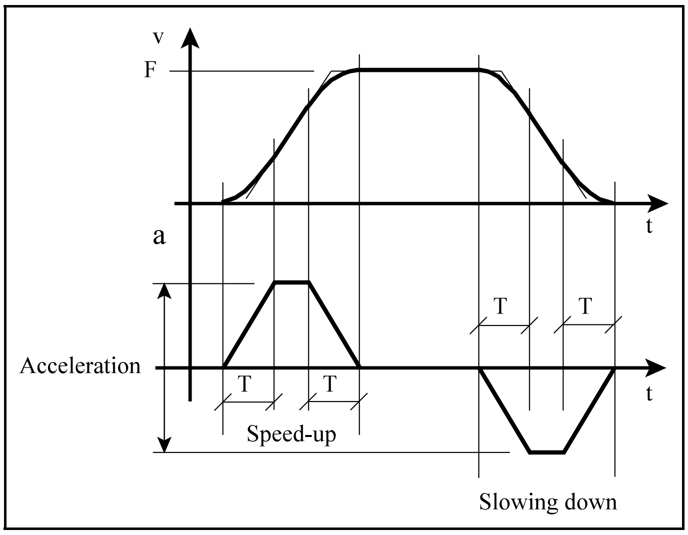


Fig. 7-2

By setting ***acceleration without jerk*** the bell-shaped acceleration can be soften further. Acceleration without jerk can be switched on by the bit position #1 JRK=1 of the parameter N0421 Acc Contr. In this case, already the leading and trailing branches of the acceleration function will be bell-shaped; in other words, there will not be step change in the first derivative (j) of the acceleration (a) either.

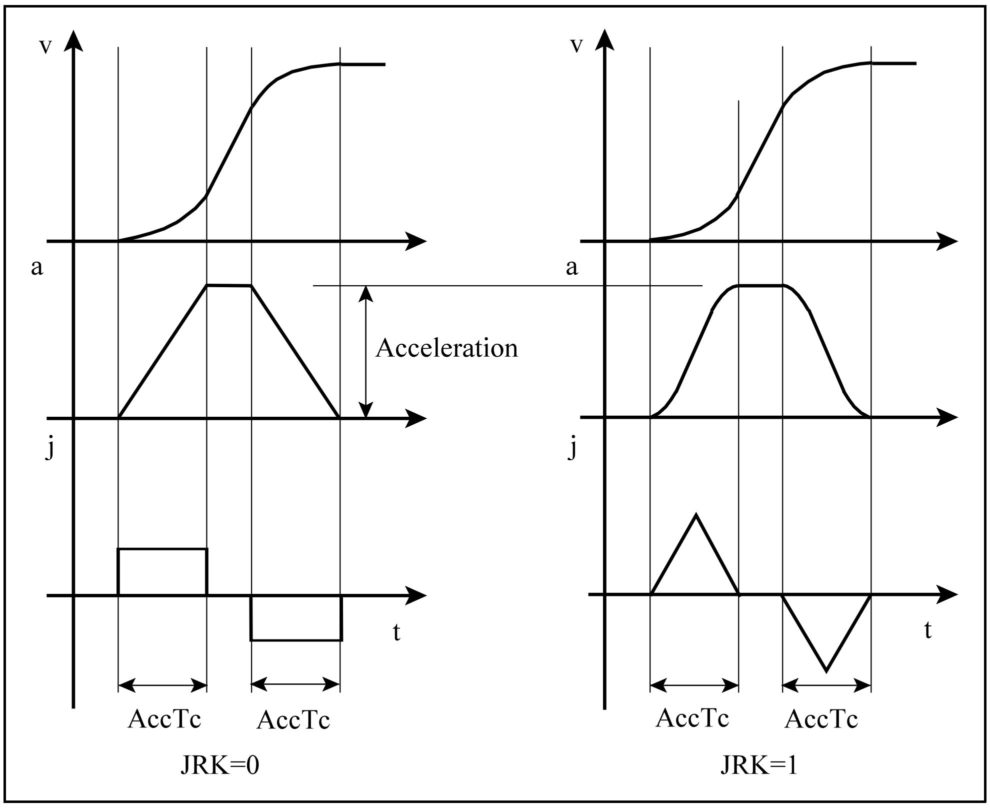


Fig. 7-3

By setting acceleration without jerk JRK=1, higher acceleration can be set on the machine, on the other hand, the start and the stop will be softer.

In the case of high-speed machining, speed feedforward has to be used in order to reach the accuracy required. In this case, bell-shaped acceleration must always be set.

In normal conditions, the control accelerates or decelerates in the following cases:
– when manual actuation is performed;
– in the case of rapid traverse positioning (G0), the motion always starts from the speed 0 at the beginning of the block, and it always decelerates to the speed 0 at the end of positioning;
– in case of feed motions (G1, G2 and G3), in the state G9 or G61, the control always decelerates to the speed 0 at the end of the block;
– the control will decelerate if motion is stopped by the button STOP, and it will accelerate if motion is started by the button START;
– the control will stop with deceleration if function is executed after motion, and at the end of the block in BLOCK BY BLOCK mode.

Acceleration to a new feed value greater than previous one will always be started by the control during execution of the block, in which the new feed is given. This process may cover several blocks too, if necessary.
Deceleration to a new feed value less than the previous on will always be started by the control in a proper previous block in such a way so that the control starts the machining with the speed programmed in that block in which the new feed is given.

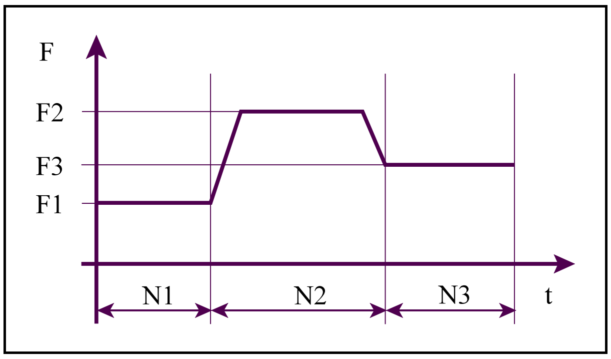


Fig. 7-4

Tangential speed changes are pre-monitored and registered by the control. It is necessary to reach the desired target speed by continuous acceleration covering execution of several blocks as well.
Either at start or at stop, reaching the desired speed may cover several blocks.

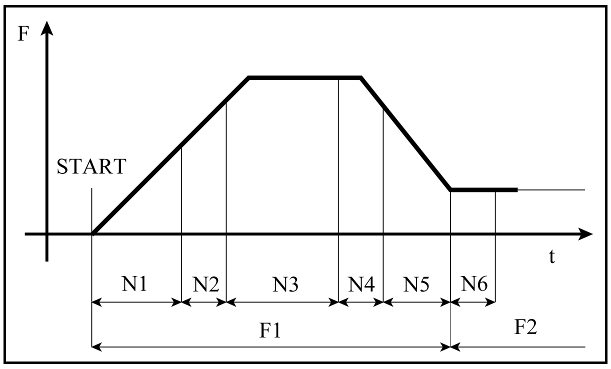


Fig. 7-5

7.1 Automatic Deceleration at Corners in the State G64

In case of continuous cutting, in the state G64, the control tries to follow the path at the programmed feedrate.
If a corner is found between two blocks, the control will have to decelerate the tangential feed.

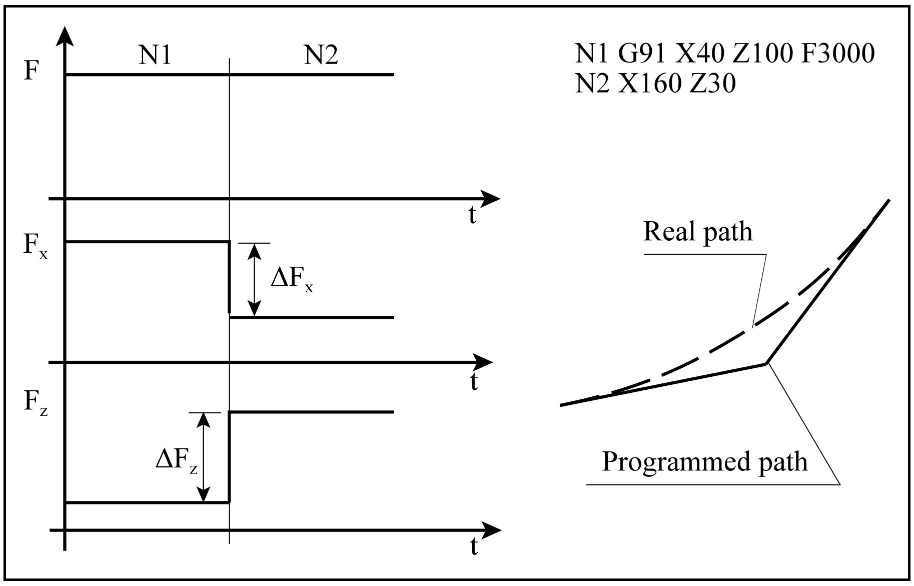


Fig. 7.1-1

If no deceleration is executed at the corner in two subsequent blocks N1 and N2 , the feed differences (ΔF_x and ΔF_z) shown in the figure will occur along the respective axes.

Detection of feedrate changes (corners) and deceleration of feed at the same time are necessary for the following two reasons:

– Feedrate changes for the axes deriving from sudden change in direction of the path may be so big, that without deceleration the drives will not be able to follow them without swinging; as a consequence, accuracy will decrease and mechanical load on the machine will increase extremely.
– If, in the course of cutting, sharp corner is to be formed, but without stop (without programming exact stop G61) since it increases cutting time, it will be necessary to decelerate too. The more the feed is decelerated, the sharper the corner will be.

The control can detect corners by monitoring either the change in direction angle of the path or the change in axis components of the feed. The method to be used for deceleration can be chosen by parameter.

Deceleration at Corners by Monitoring the Change in the Direction Angle of the Path

At the bit position #0 FDF=0 of the parameter N0306 Feed Control, deceleration will be executed by monitoring the change in the direction angle of the path.

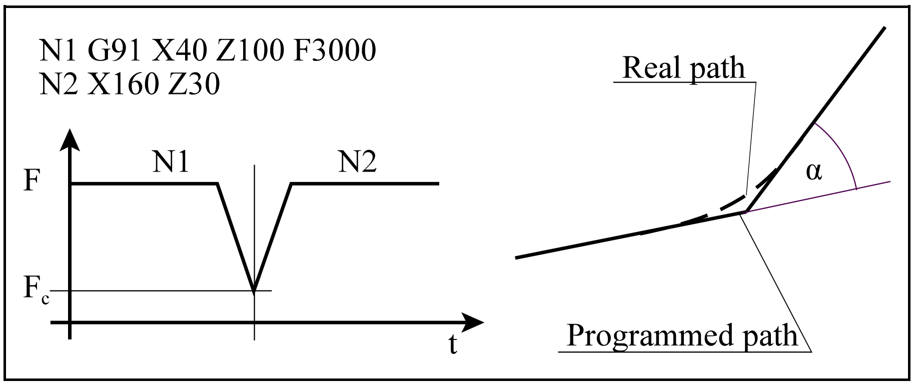


Fig. 7.1-2

If, at the meeting point of the blocks N1 and N2 shown in the figure, ***the value of the angle α exceeds*** the value enabled in parameter, the control will decelerate the feed to the value F_c.
It is the parameter N0307 Crit A Diff , in which the value of the critical angle can be set in degree: α=Crit A Diff. The value in the parameter N0308 Feed Corn determines the value of corner feed to which the control has to decelerate in the case of exceeding the critical angle: F_c=Feed Corn.
The greater the values of the critical angle and the corner feed are, the faster the machining will be, but the bigger the load acting on the machine will be and the more rounded the corner will be. *This set-up is not appropriate for high-speed machining.*

Deceleration at Corners by Monitoring the Change in Axis Components of the Feed

At the bit position #0 FDF=1 of the parameter N0306 Feed Control, deceleration will be executed by monitoring the change in axis components of the feed.

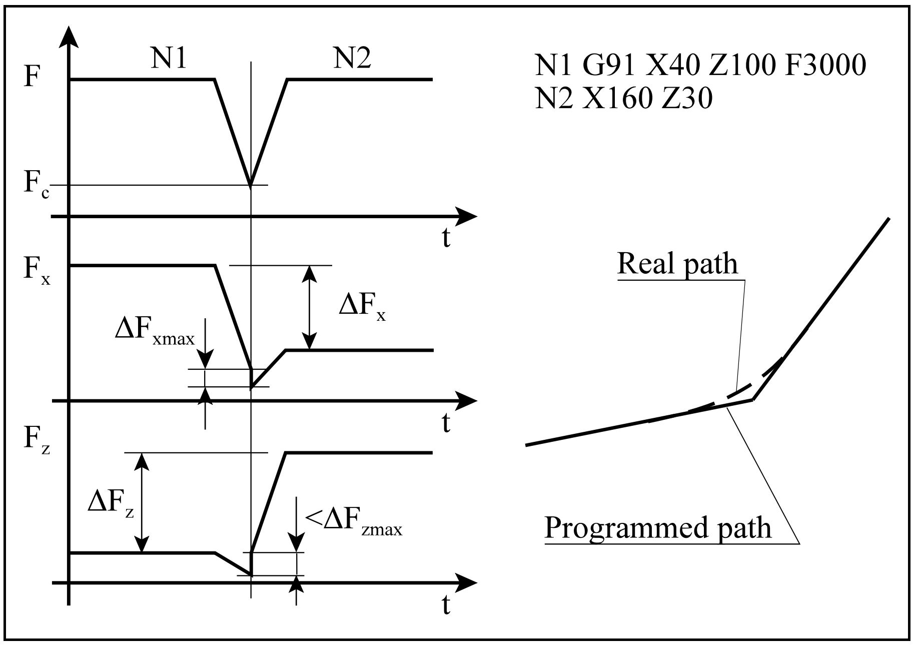


Fig. 7.1-3

If, at the meeting point of the blocks N1 and N2 shown in the figure, the change in axis components of the feed ΔF_x, ΔF_z exceeds the maximum value enabled in parameter, the control will decelerate the tangential feed F to the value F_c.
The control decelerates feed in such a way, so that the value of feed change does not exceed, on none of the axes, the critical feed difference (ΔF_{xmax}, ΔF_{zmax}) enabled for the given axis in parameter; the critical feed difference for a given axis can be specified in the parameter N0309 Crit F Diff : ΔF_{xmax}=Crit F Diff$_x$, ΔF_{zmax}=Crit F Diff$_z$.
The greater the value of the critical feed difference is, the faster the machining will be, but the bigger the load acting on the machine will be and the more rounded the corner will be.
The control bounds the value of the preset critical feed difference from above, on the basis of acceleration set-up.
This set-up must be applied for high-speed machining.

7.2 Limiting the Normal Direction Accelerations

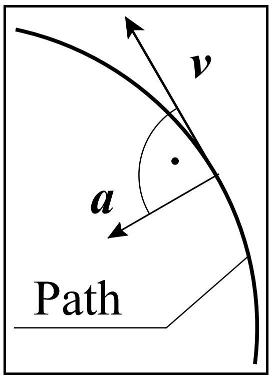


Fig. 7.2-1

In the course of machining, the control keeps the feedrate constant along the tangent of the path (in tangential direction). As a consequence, acceleration components do not come into being in tangential direction. The situation is different in normal direction (in the direction perpendicular to the path and the velocity). The axial components of the normal direction acceleration may exceed the value permissible for the given axis. In order to avoid this, the feedrate along the path must be limited in proportion to the curvature of the path.
The maximum permissible value of the normal direction acceleration can be set in the parameter N0402 Normal Acc.

Limiting the Normal Direction Acceleration in case of Circular Arcs
In the course of machining circular arcs, the control limits the value of the feedrate F according to the formula

$$F = \sqrt{a \cdot r}$$

where:
    a: the lesser one among the acceleration values set for axes participating in circular interpolation ;
    r: the radius of the circle.
The circular interpolation will already be started using the rate calculated in this way.
The control does not decrease the feedrate under the value specified in the parameter N0310 Circ F Min, independently of formula above.
☞ ***Warning!***
This function should not be confused with automatic feed override in states G41and G42, in the course of machining inner circular arcs.

Limiting the Normal Direction Acceleration in case of Other Interpolations

Though ***in the case of linear interpolation***, the given line segment (path) does not have curvature, therefore there is no normal direction acceleration component either, but this is true for long straight line segments only. If a ***path is made up of minute straight line segments***, as it is usual in manufacturing tools, the curvature of the path resulted in such a way can be significant, and feed will have to be decreased, as it is illustrated in the figure below:

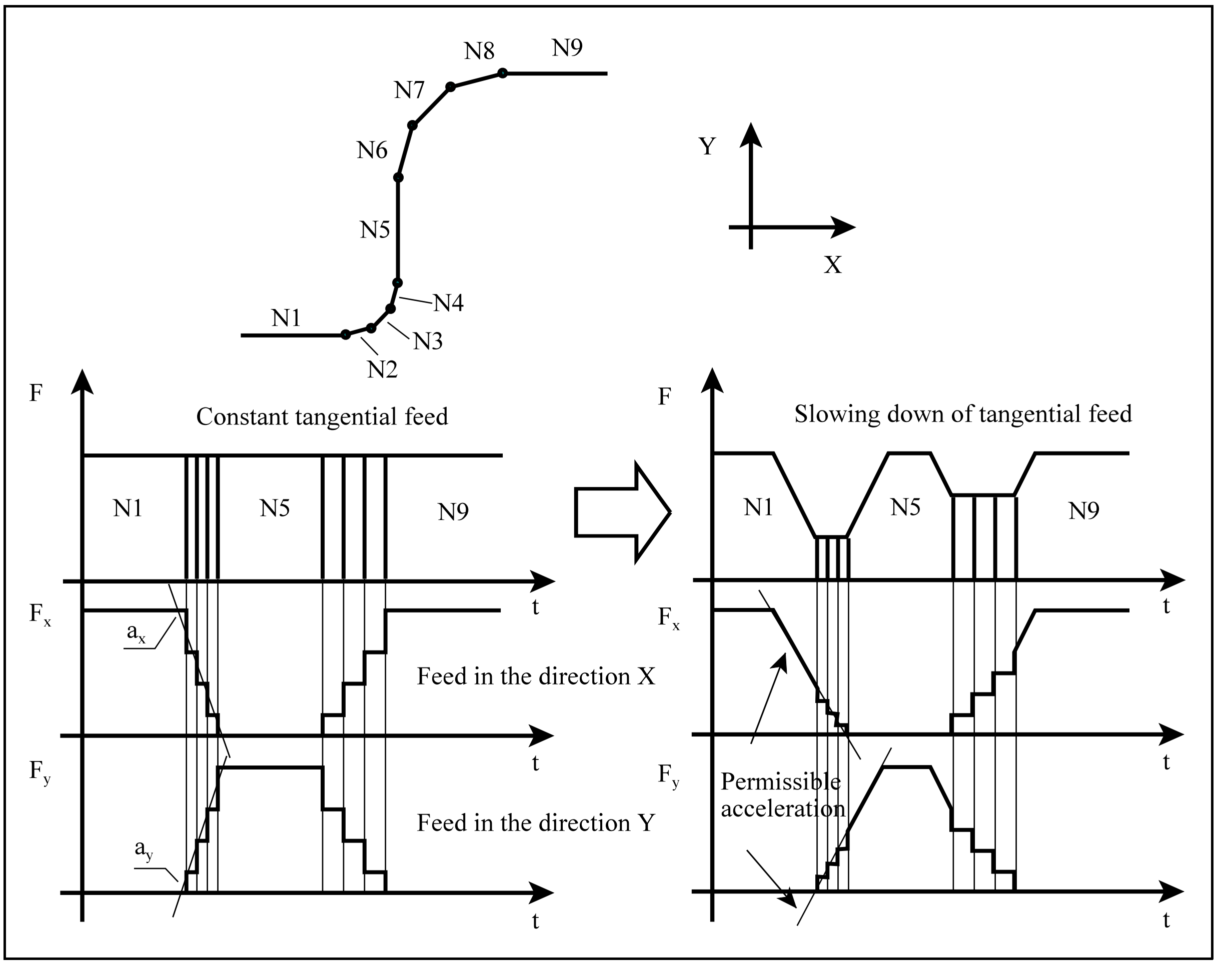


Fig. 7.2-2

In the blocks N2, N3, N4 and N6, N7, N8, the path is made up of minute straight line segments. If the value of tangential feed is kept constant (the left part of the figure above), the gradient of speed change (acceleration in the normal direction) on the axes X and Y may exceed the value permissible for the given axis, because of geometry (direction change) of the path.

For this reason, the control scans the path block by block in order to be able to limit normal direction accelerations. Where, from the geometry, the acceleration components on given axes are greater than the permissible values, the tangential speed must be decreased. The graphs on the right part of the figure shows, how the measure of speed change (the normal direction acceleration) decreases on the given axes in proportion to deceleration of tangential feed.

In the case of the ***smooth interpolation*** G5.1 Q2, the control also examines the normal direction accelerations derived from the path curvature, and it decreases the feed, if necessary.

7.3 Limiting the Acceleration Step Change (Jerk)

On the certain sections of the path, sudden step change, jerk may occur that causes swings, loads the machine mechanically, and appears on the surface machined. This is the case, when in the course of machining a tangent circle follows a straight line segment or a circle arc is followed by a tangent straight line.
The purpose of this function is that the control limits the value of acceleration step change in the transition point by decreasing the feed.

Limiting the Acceleration Step Change at the Beginning and at the End of Circle Blocks
In the case of entering from a straight line segment into a tangent circle arc at a feedrate F, the value of the acceleration step change can be calculated using the formula

$$a = \frac{F^2}{r}$$

where:
a: the value of the acceleration step change,
r: the radius of the circle.
For example, if the machine, at the feedrate F6000, enters into a circle arc with radius of 10 mm, as it is illustrated in the figure below, the value of the acceleration step change on the axis Y will be:

$$a = \frac{\left(\frac{6000}{60}\frac{mm}{\sec}\right)^2}{10mm} = 1000\frac{mm}{\sec^2}$$

In order that the value of the acceleration step change will not be greater than 250mm/sec^2, the feedrate has to be decreased to the value F=50*60=3000 mm/min, using the equation above.

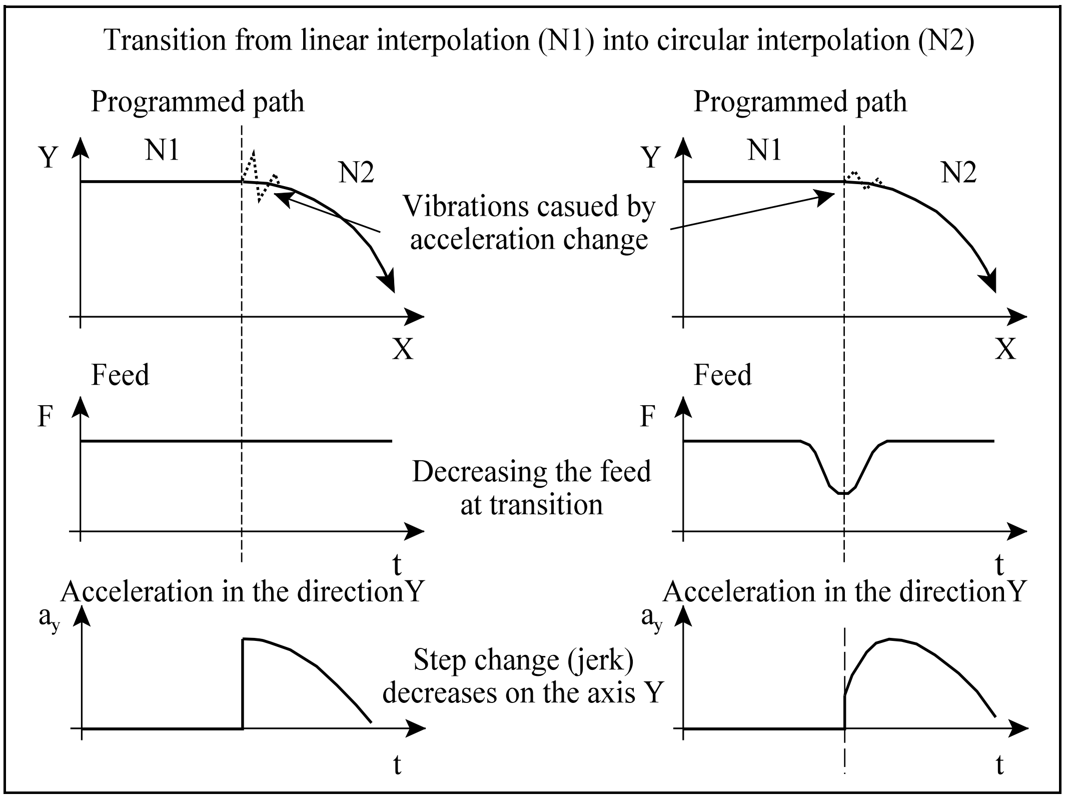

Transition from linear interpolation (N1) into circular interpolation (N2)
Programmed path
Y
N1
N2
X
Programmed path
Y
N1
N2
X
Vibrations casued by acceleration change
Feed
F
t
Feed
F
t
Decreasing the feed at transition
Acceleration in the directionY
a_y
t
Acceleration in the directionY
a_y
t
Step change (jerk) decreases on the axis Y

Fig. 7.3-1

In case of circle arcs, the acceleration step change can be limited in the parameter N0404 Acc Diff Circ.

Limiting the Acceleration Step Change in Straight Line Blocks following Each Other

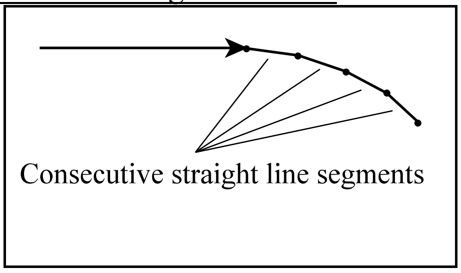


Fig. 7.3-2

If the path is composed of long straight line segments, the measure of acceleration change will be negligible. In this case, the change in axis components of the feed could limit the feed.
The situation will be different if the path is made up of minute straight line segments. In this case, there could take place the situation when the feed change on the given axes is small between two straight line segments; because of this the interpolator does not limit the feed, but the values of acceleration step change on the given axes are great. In such cases, the feed must also be limited depending on the permissible acceleration step change.
In case of straight line segments following each other, the acceleration step change can be limited in the parameter N0403 Acc Diff .

8 Dwell (G4)

Using the command

```
        G94 G4 P....
```

dwell time can be programmed in second.
Using the command

```
        G95 G4 P....
```

dwell time can be programmed in number of spindle revolutions.
The accuracy of P is 15 decimal digits.

At the bit state #1 SEC=1of the parameter N1337 Execution Config , dwell is always counted in second, even if in the state G95.
Dwell always means the programmed delay of execution of the next block. It is a non-modal function.

9 Reference Point

The reference point is the point of the axis where the measuring system sends out position 0.
On the axes equipped with incremental measuring devices this point has to be found. This process is called reference point return.
Measuring the workpiece coordinate systems and positioning to an absolute position can be done after finding the reference point. The parametric end-positions and programmed travel limits will be effective after reference point return only.

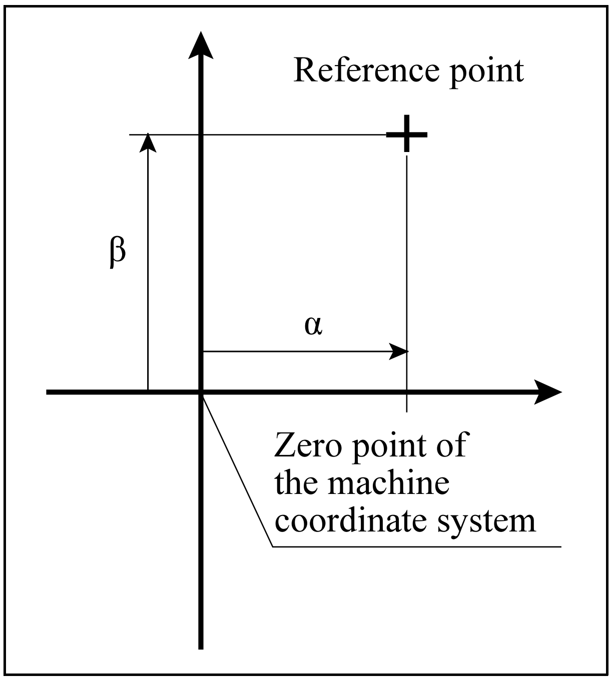


Fig. 9-1

Positions are recorded by the control not as values relative to the reference point, but as values in the machine coordinate system.
The zero point of the ***machine coordinate system*** is specified by the builder of the machine and it is a significant point on the machine tool.
The control records the change positions, center of rotation of the rotary axes etc. in the machine coordinate system. All the compensations (for thread pitch, straightness etc.) of the machine measuring system are recorded in the machine coordinate system, too.

The ***position of the reference point*** is recorded by the control ***in the machine coordinate system too***, and this position is set by the builder of the machine in parameter.

If the axis is equipped with ***incremental measuring system***, return to the reference point must be executed after power-on. In the course of reference point return, the slides run onto a switch in the direction specified in parameter, and then, coming down from it, they look for the zero pulse of the measuring system and record existence of the reference point. The value given in parameter will be the position in the machine coordinate system.

If the axis is equipped with ***distance-coded measuring system***, return to the reference point must be executed after power-on. In the course of reference point return, the slides start in the direction specified in parameter, and then they look for two zero pulses and record existence of the reference point. The position of the zero pulse found as second one will be the position in the machine coordinate system.

If the axis is equipped with ***absolute measuring system***, return to the reference point does not have to be executed after power-on.

The reference point is also the point ***in the cases of both distance-coded and absolute measuring systems*** where the measuring system sends out position 0. Generally, this point is ***not***

within the working range of the machine. For this reason, the builder of the machine, by the use of parameterization, shifts this point into the working space, e.g. near to the positive endpoint, and then he measures this shifted reference point to the origin of the machine coordinate system. It is necessary, for example in the case, when the instruction G28 is to be used in the part program, for example for positioning together with tools.

9.1 Automatic Reference Point Return (G28)

The instruction

G28 v

will return the axes defined by the vector v to the reference point. There are two phases of the motion.

Phase 1

At first, the control, taking the coordinates given ***in the actual workpiece coordinate system*** into account as ***intermediate point***, executes rapid traverse along the axes defined by the vector v to the intermediate point specified by the vector v. The specified coordinate values can be either absolute or incremental ones. When the intermediate point is reached, the planar tool radius compensation is deleted.

Phase 2

Then, from the intermediate point, reference point return will be executed simultaneously on the axes specified by the vector v.

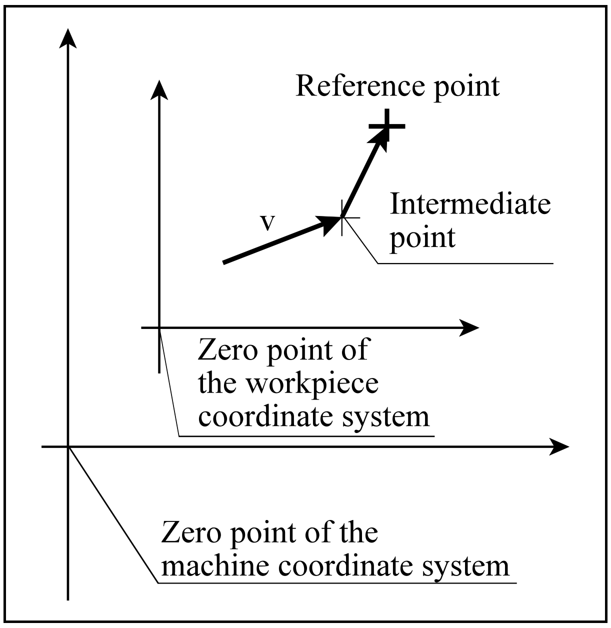

Reference point
v
Intermediate point
Zero point of the workpiece coordinate system
Zero point of the machine coordinate system

Fig. 9.1-1

If reference point return on a given axis did not occur yet, it would occur according to the mode determined by the manual reference point return. In this case,

– if the axis is equipped with ***incremental measuring system***, the reference point position specified in parameter will be the machine position at the end of the motion;

– if the axis is equipped with ***distance-coded measuring system***, the position of the second zero pulse will be the machine position at the end of the motion.

If reference point return on a given axis occurred already, or the axis is equipped with absolute measuring system, the axis executes rapid traverse to the reference point position specified in the machine coordinate system.

The code G28 is not a modal one.

For example:

```
G90 G28 X100 Z50 (the intermediate point: X=100, Z50)
```

If the position X is X=20 and the position Z is Z=50:

```
G91 G28 X100 Z50 (the intermediate point: X=120, Z=100)
```

☞ *Note*:

– If there is no valid reference point yet, incremental values must be assigned to the intermediate coordinates of v existing in the command G28.

9.2 Return to the 2nd, 3rd and 4th Reference Point (G30)

Three additional significant points called ***2nd, 3rd and 4th reference point*** can be specified ***in parameter***, in the ***machine coordinate system***.
These reference points are used on the machine to store change positions, e.g. tool change position, pallet change position etc.
Motion to these change positions is permitted after execution of reference point return only.

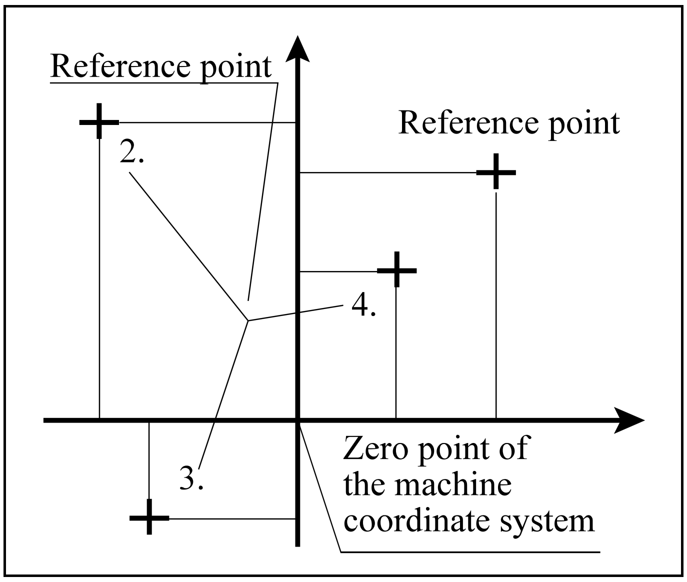


Fig. 9.2-1

The series of instructions

G30 v P

sends the axes coordinates of which are specified at the addresses of the vector v, to the reference point specified at the address P.

P2: the 2nd reference point
P3: the 3rd reference point
P4: the 4th reference point

There are two phases of the motion as it was the situation in the case of the instruction G28.
At first, the control, taking the coordinates specified by the vector v into account as intermediate point, executes rapid traverse of linear motion ***to the intermediate coordinates*** specified by the vector v. The specified coordinate values can be either absolute or incremental ones. The motion is always executed in the actual coordinate system. When the end point of the linear motion is reached, the planar tool radius compensation vector is deleted.
In the second phase, the axes specified by the vector v execute rapid traverse from the intermediate point ***to the reference point selected at the address P***.
Travelling to the reference point is carried out by ignoring compensation vectors (length, offset, 3D radius); they do not have to be deleted before issuing the instruction G30, but the control will implement them in the course of programming further motions. The planar tool radius compensation resets automatically in the first motion block.
It is not a modal code.

The instruction G30 v P1moves the machine to the reference point, its effect and the effect of the instruction G28 is the same.

For example:

```
     G90 G30 X100 Z50 P3       (the intermediate point X=100, Z50
                               moves to the P3)
```

If the position X is X=20, and the position Z is Z=50:

```
     G91 G30 X100 Z50 P4       (the intermediate point X=120,
                               Z=100 moves to the P4)
```

10 Coordinate Systems and Plane Selection

In the program, a position the tool is to be moved to is specified by coordinate data. When two axes (X and Z) are used, the position of the tool is expressed by two coordinate data: X____ and Z____.

As many axes are there on the machine, so many coordinate data express the tool position. The coordinate data have to always be interpreted in a given coordinate system.

The control differentiates the following three coordinate system:

1. The machine coordinate system
2. The workpiece coordinate system
3. The local coordinate system

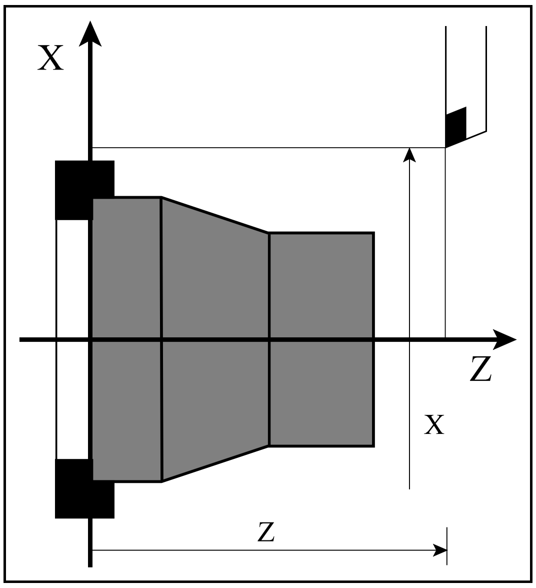


Fig. 10-1

10.1 Machine Coordinate System

Positions are recorded by the control not as values relative to the reference point, but as values in the machine coordinate system.

The zero point of the machine coordinate system is specified by the builder of the machine and it is a significant point on the machine tool.

The control records the change positions, center of rotation of the rotary axes etc. in the machine coordinate system. All the compensations (for thread pitch, straightness etc.) of the machine measuring system are recorded in the machine coordinate system, too.

The ***position of the reference point*** is recorded by the control ***in the machine coordinate system too***, and this position is set by the builder of the machine in parameter.

The position of the machine coordinate system cannot be changed by the use of any instruction or offset.

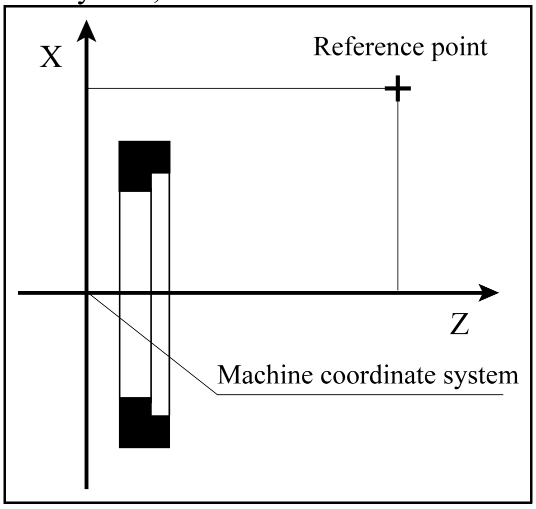


Fig. 10.1-1

10.1.1 Positioning in the Machine Coordinate System (G53)

The instruction

G53 v

moves the tool to the point of coordinate v in the machine coordinate system.

– Independently of the state of G90 and G91, coordinates of v are always interpreted as ***absolute coordinates***.

– After the address of the coordinates or in the case of using address U, V and W for incremental specification, the operator I will send the error message '2097 Illegal incremental moving on ... axis'.

– Motion is always ***rapid travel***, similarly to the case of G00.

– Positioning is always executed by ignoring the selected and set tool compensations (length and radius).

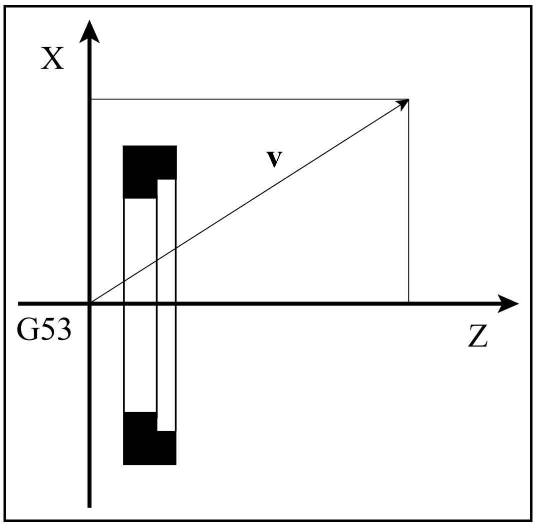


Fig. 10.1.1-1

The instruction G53 can be executed after reference point return only. The instruction G53 is a ***one-shot*** instruction , it is effective in that block only in which it is given.

An example: The effect of the instruction

```
    G53 X200 Z20
```

is motion to the specified point in the machine coordinate system.

☞ ***Warning!*** *The instruction G53 suspends the read ahead of the blocks(buffering). Therefore, the instruction G53 in itself without coordinate specification can also be used to suspend reading ahead of the blocks, that is to empty the buffer.*

10.2 Workpiece Coordinate Systems

The coordinate system in which the part program has to be written is named workpiece coordinate system. The control stores the origin of the workpiece coordinate system relatively to the machine coordinate system.

The origin of the workpiece coordinate system is fixed to an appropriate point on the workpiece. This point can be the axis of rotation in the direction X and the contact surface of the chuck or the front of the workpiece in the direction Z, in case of turning; but in case of milling, it can be one of the corners of the workpiece, the center point of a hole or a collar etc. Setting can be carried out in the following ways:

within the machine by manual measurement, or by measurement using probe; or

outside the machine.

In the latter case, the values measured outside have to be input in the memory of the control. Data input can be done manually, or from program by the use of NC instructions.

Relationship between the Workpiece Coordinate Systems and the Channels

Offsets of the workpiece coordinate systems refer to the given axes. Since each axis is assigned in parameter to a given channel, therefore, each channel has different workpiece offset table.

If one or more axes are interchanged between two channels, the axes will take their zero point offset away into the new channel. In such cases, after axis interchange, it is practical to calla new workpiece coordinate system together with absolute positioning, and to continue the machining after that.

10.2.1 Selecting the Workpiece Coordinate System (G54...G59)

In basic version, 6 different workpiece coordinate systems are stored by the control. The ***offsets*** of the workpiece coordinate systems ***relative to the origin of the machine coordinate system*** must be given for each axis separately.

The case, when there is no common zero point offset, illustrated in the figure below.

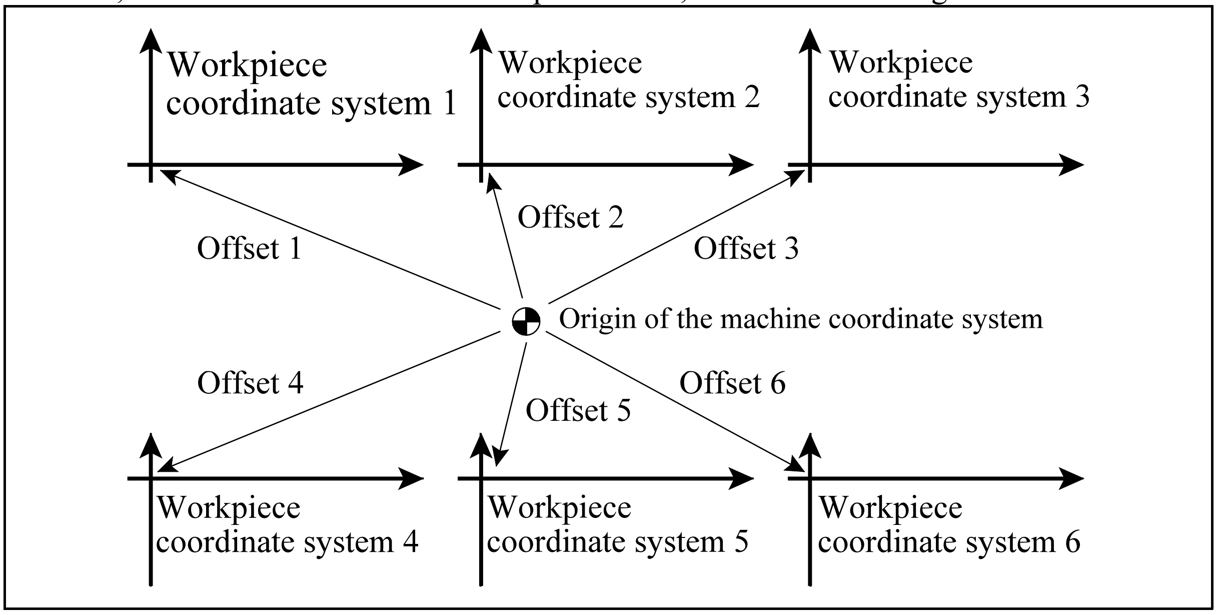

Workpiece coordinate system 1
Workpiece coordinate system 2
Workpiece coordinate system 3
Offset 2
Offset 1
Offset 3
Origin of the machine coordinate system
Offset 4
Offset 6
Offset 5
Workpiece coordinate system 4
Workpiece coordinate system 5
Workpiece coordinate system 6

Fig. 10.2.1-1

All the workpiece coordinate system can be offset relative to the origin of the machine coordinate system. ***A common zero point offset will shift the origins of all the workpiece coordinate systems*** relative to the machine coordinate system.

The following figure shows the case, when there is common zero point offset.

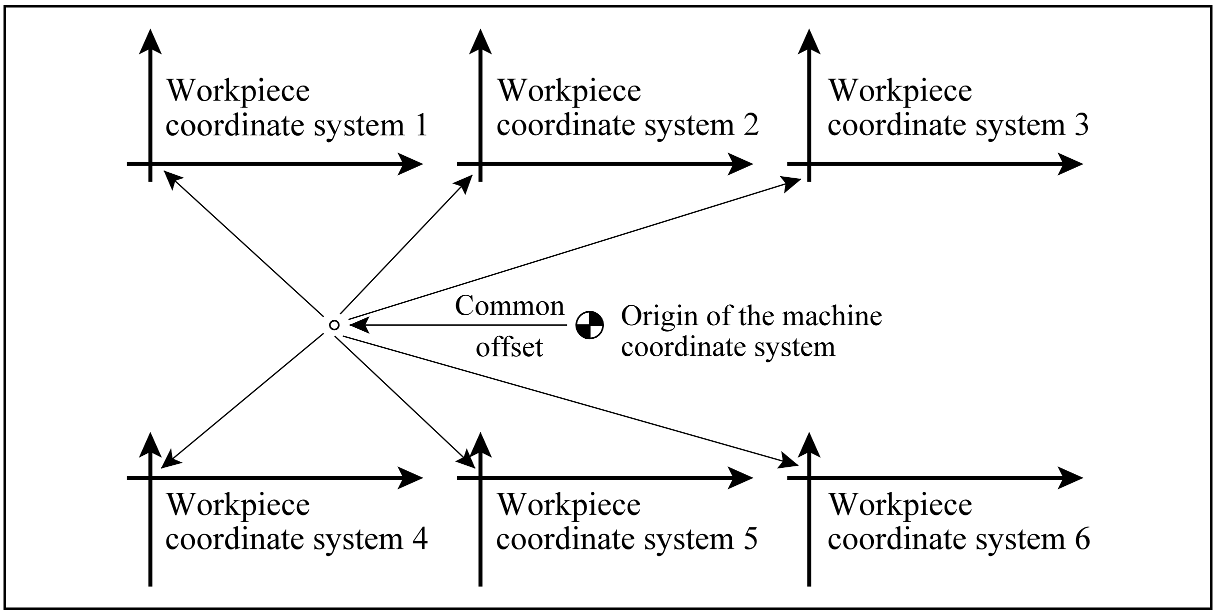

Workpiece coordinate system 1
Workpiece coordinate system 2
Workpiece coordinate system 3
Common offset
Origin of the machine coordinate system
Workpiece coordinate system 4
Workpiece coordinate system 5
Workpiece coordinate system 6

Fig. 10.2.1-2

The instructions G54 ... G59 are those, by the use of which the selection can be done from among the various workpiece coordinate systems.

G54: Workpiece coordinate system 1
G55: Workpiece coordinate system 2
G56: Workpiece coordinate system 3
G57: Workpiece coordinate system 4
G58: Workpiece coordinate system 5
G59: Workpiece coordinate system 6

These functions are modal ones.
After power-on, reference point return, reset or program end, the coordinate system G54 will be selected.
The absolute coordinate data of the interpolation blocks are taken into account by the control in the actual workpiece coordinate system.
For example, in the case of the instruction

```
G56 G90 G00 X80 Z60,
```

positioning to the point

X=80, Z=60

of the workpiece coordinate system 3 will be executed.

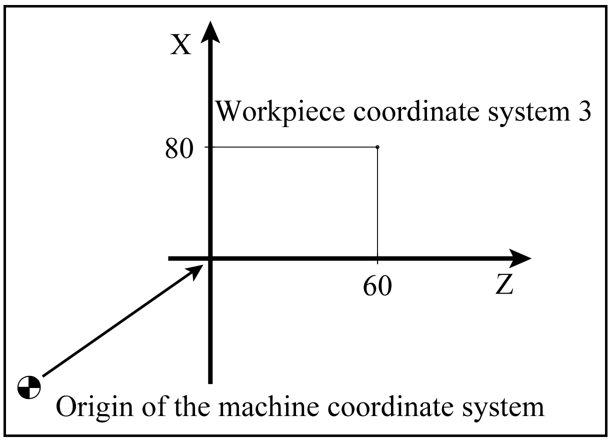

X
Workpiece coordinate system 3
80
60
Z
Origin of the machine coordinate system

Fig. 10.2.1-3

After changing the coordinate system, the position of the tool will be displayed in the new coordinate system. For example, as it is shown in the figure, the ***offset*** of the workpiece coordinate system ***G54*** in the machine coordinate system is

X=260 Z=80.

The ***offset*** of the workpiece coordinate system ***G55*** in the machine coordinate system is

X=140, Z=180.

The ***position of the tool*** in the coordinate system ***G54*** X', Z' is

X'=140, Z'=90.

By the effect of the instruction ***G55, the position of the tool*** in the coordinate system X", Z" will be interpreted as

X"=260, Z"=–50.

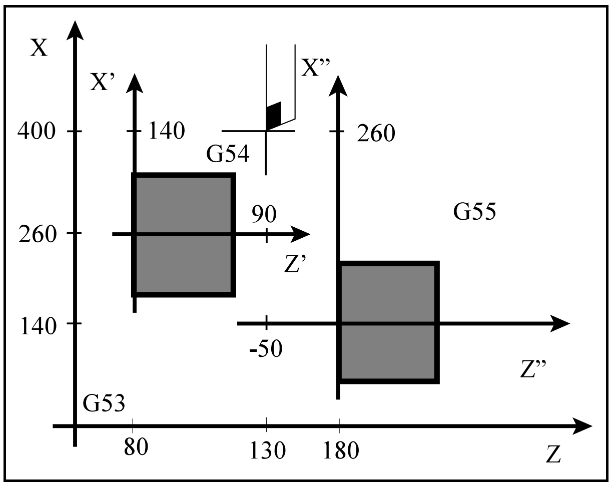


Fig. 10.2.1-4

10.2.2 Selecting the Additional Workpiece Coordinate Systems (G54.1 P)

Optionally, 99 further workpiece coordinate systems can be used in the control. These coordinate systems are augmentations to the 6 basic coordinate systems so they are called additional workpiece coordinate system.

The ***common zero point offset*** shifts the additional workpiece coordinate system too. The additional workpiece coordinate systems ***can be rotated*** just as the basic coordinate systems can be.

The instruction

G54.1 Pp

is used for selection of a additional workpiece coordinate system, where the number of the additional workpiece coordinate system can be specified at the address P:

P = 1, 2, ..., 99

It is a modal function.

☞ ***Warning!*** The address P can serve various purposes. Therefore, the use of the address P in the block must be unequivocal:

```
G0 G54.1 P16 X100 Z20 M98 P1 (NOT CORRECT! Two Ps in one
block)
G0 G54.1 P16 X100 Z20 (unequivocal)
```

10.2.3 Compensating the Angular Position of the Workpiece

Compensation of the angular position of the workpiece is needed when the workpiece cannot be aligned parallel to the main axes. Hereafter, this compensation is called ***misalignment compensation***.

Interpretation of misalignment compensation

In the general zero point offset table below, the columns have the following meanings:

X, Y, Z: the zero point offset along the 3 main axes,

α, β, γ: the angles of the misalignment compensation, namely, α in the plane G19, β in the plane G18, γ in the plane G17.

	X	**Y**	**Z**	**G17(γ)**	**G18(β)**	**G19(α)**
G54						
...						
G54.1 P						
...						

The rotations are always about the main axes. The order of rotations is as follows:
Rotation 1: rotation about X axis at an angle α,
Rotation 2: rotation about Y axis at an angle β,
Rotation 3: rotation about Z axis at an angle γ.

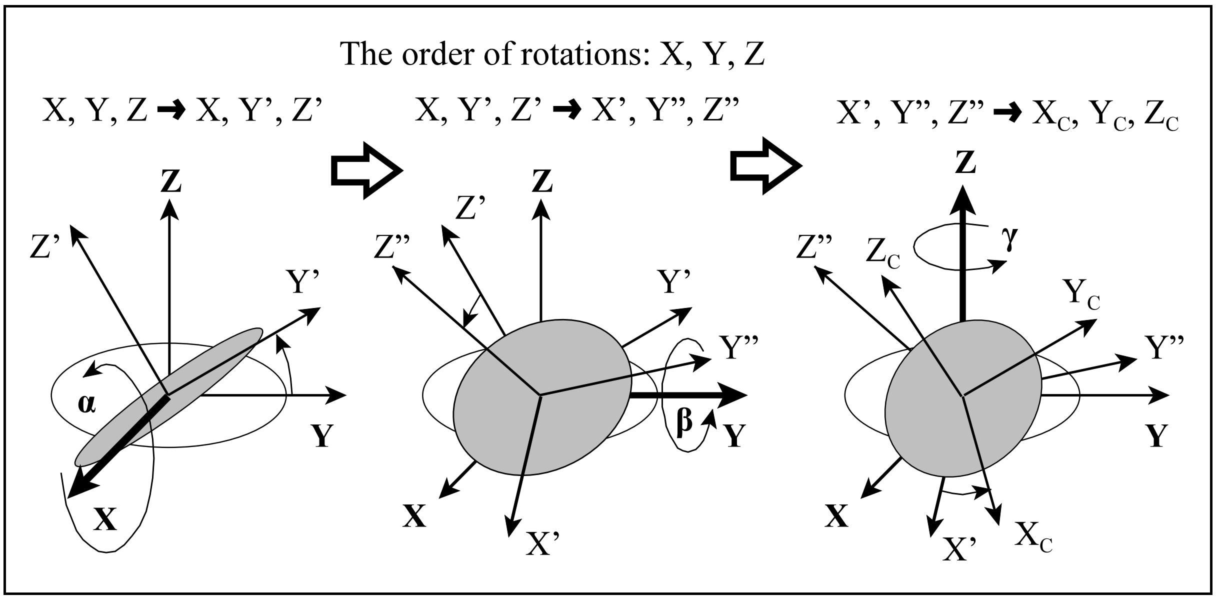


Fig. 10.2.3

The direction of α, ß and γ is always the direction of the rotational transformations about the corresponding main axis, according to the right-hand rule.

The misalignment compensation transformations ***are always taken into account in the current workpiece coordinate system, in the case of motion commands***.
It never applies to commands on motion to a machine position (G53, G28, G30).

In the case of manual moving, the controller takes into account the switch 'In accordance with misalignment compensation' (CP_WMCAXF PLC flag), i.e. it applies rotation about all three axes.

If the endpoint is **v**[x, y, z], rotation takes place in accordance with the equation

v'=$\mathbf{M}_Z(\gamma)$*$\mathbf{M}_Y(\beta)$*$\mathbf{M}_X(\alpha)$*v

where:

v'[x', y',z'] are the coordinates rotated, and

$\mathbf{M}_X(\alpha)$, $\mathbf{M}_Y(\beta)$, $\mathbf{M}_Z(\gamma)$ are the matrices of rotations about the X, Y and Z axes:

$$M_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad M_Y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad M_Z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Applying misalignment compensation for lathe

The figure below shows an example for applying misalignment compensation. The sub-spindle S2 in the turret T1 reaches the rear tool group T2 so that the its axis of rotation S2 makes an angle ß with the axis Z. In this case, the misalignment compensation can be set in the column G19 β. Henceforth, the program can be written using the Z'X' coordinate system.

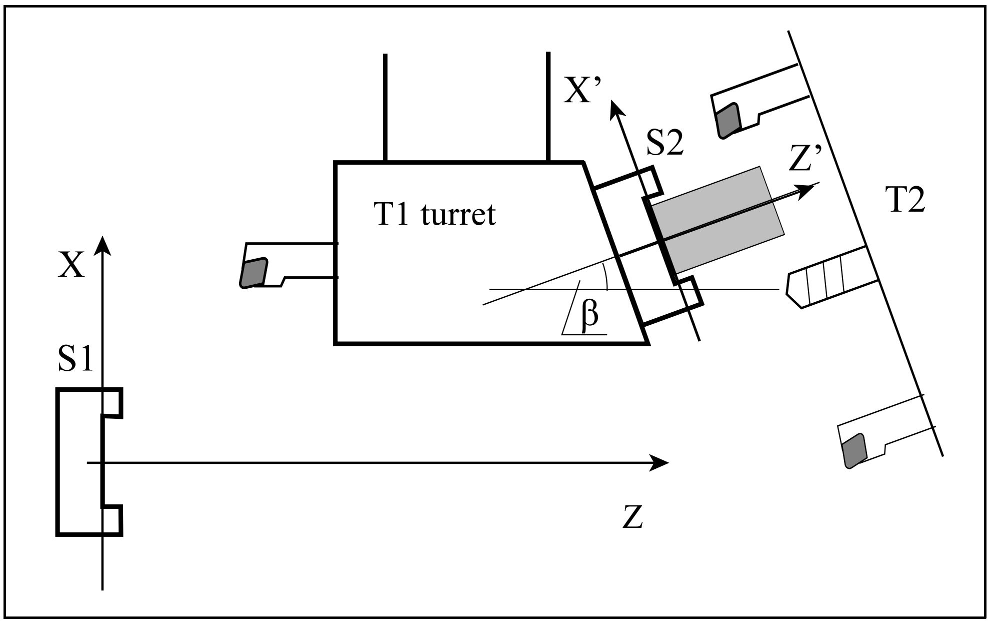


Fig. 10.2.3

10.2.4 Setting the Offset of the Workpiece Coordinate Systems (G10 L2)

Offset, rotation and common zero point offset of the workpiece coordinate systems can also be set using program instruction.

Setting is executed by the instruction

G10 L2 P v I J K,

where

P = 0	setting the common zero point offset;
P = 1...6	selecting the workpiece coordinate system G54, ..., G59;
v (X, Y, Z, ...):	offset values for the axes.

Axis offsets are always entered as Cartesian values; length data in mm or in inch, angle data in degree.

I:	γ the angle of rotation in the plane G17;
J:	β the angle of rotation in the plane G18;
K:	α the angle of rotation in the plane G19.

The angle of rotation must always be specified in degree.
For common zero point offset, rotation cannot be specified.
G10 is a one-shot instruction.
In the absolute data setting command state **G90,** the value written at the coordinate addresses or at the address I, J and K will be put in the appropriate offset register.
In the incremental data setting command state **G91** or in the case of using the operator I, the data written at the addresses will be added to the content of the appropriate offset register. The operator I can be used for coordinate addresses only, but not for the address I, J and K.

10.2.5 Setting the Offset of the Additional Workpiece Coordinate Systems (G10 L20)

Offset and rotation of the additional workpiece coordinate systems can also be set using program instruction.
Setting is executed by the instruction

G10 L20 P v I J K,

where

P = 1...n	selecting the workpiece coordinate system G54.1 P1, G54.1 P2, ..., G54.1 Pn;
v (X, Y, Z, ...):	offset values for the axes.

Axis offsets are always entered as Cartesian values; length data in mm or in inch, angle data in degree.

I:	γ the angle of rotation in the plane G17;
J:	β the angle of rotation in the plane G18;
K:	α the angle of rotation in the plane G19.

The angle of rotation must always be specified in degree.
G10 is a one-shot instruction.
In the absolute data setting command state **G90,** the value written at the coordinate addresses or at the address I, J and K will be put in the appropriate offset register.
In the incremental data setting command state **G91** or in the case of using the operator I, the data written at the addresses will be added to the content of the appropriate offset register. The operator I can be used for coordinate addresses only, but not for the address I, J and K.

10.2.6 Creating a New Work Coordinate System (G92)

The command

G92 v

establishes a new workpiece coordinate system in such a way that a designated point, e.g. the tool tip will be the point of coordinate v of the new workpiece coordinate system. Afterwards, any following absolute command will have to be interpreted in this new workpiece coordinate system, and positions will also be displayed in this coordinate system. The coordinates specified in the command G92 will always be interpreted as absolute Cartesian values.

For example, if the tool is at the point of the coordinates

X=200, Z=150,

in the actual X, Z coordinate system, the command

```
G92 X120 Z90
```

creates the new X', Z' coordinate system in which the tool will be at the point of the coordinates

X'=120, Z'=90.

The axis components of the offset vector **v'** between the coordinate systems X, Z and X', Z' will be the following:

v'_x=200-80=120, and

v'_z=150-90=60.

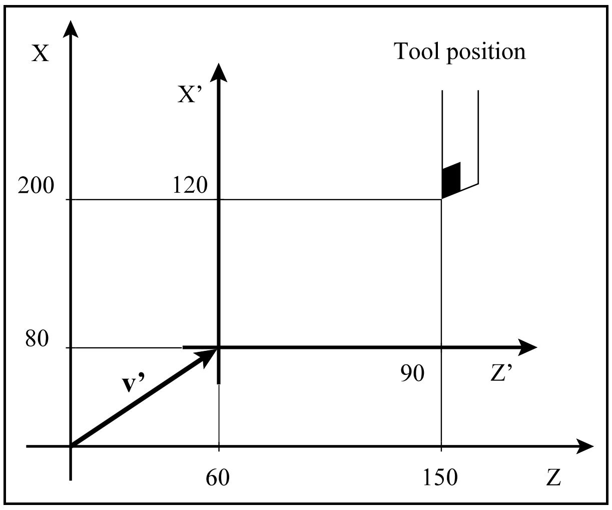


Fig. 10.2.6-3

The command G92 is valid in each workpiece coordinate system, i.e. the offset **v** calculated in one of them will be taken into account in the other ones too.

The offset of the workpiece coordinate system set by the command G92 will be cancelled by power-on, at the end of the program and by reset.

The command ***G92 cancels the tool radius compensation vector***, it will not be included in calculation of the offset.

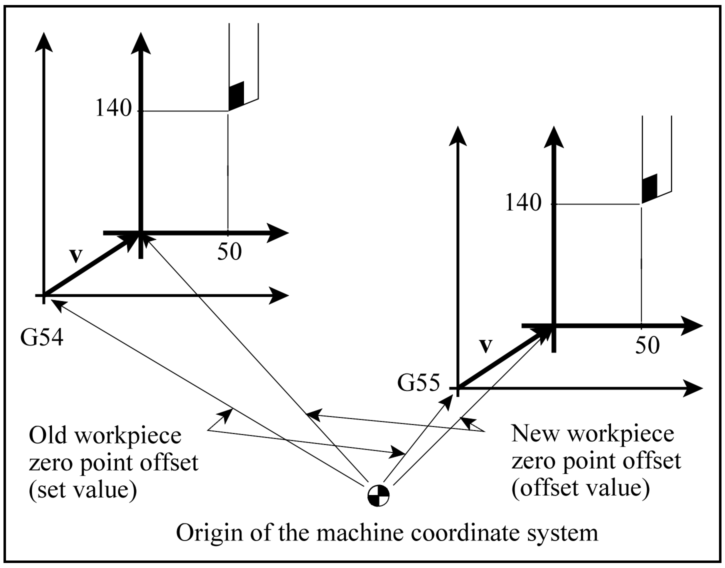

140
v
50
G54
140
v
50
G55
Old workpiece zero point offset (set value)
New workpiece zero point offset (offset value)
Origin of the machine coordinate system

Fig. 10.2.6-4

The command ***does not cancel the length compensation vector***, the offset will always be calculated for the position of the tool tip.

The command ***G92 cancels*** the local coordinate system's ***offsets*** programmed by the command ***G52*** on the axes that are given in the command.

During motion, if the valid workpiece coordinate system is rotated, the offset vector specified in the command G92 will be taken into account as rotated vector.

10.3 Local Coordinate System (G52)

When writing part programs, it is sometimes more convenient to specify the coordinate data in a so-called local coordinate system instead of the work coordinate system.

The command

G52 v

creates a local coordinate system.

If the coordinate **v** is specified **as absolute value**, the origin of the local coordinate system will coincide with the workpiece coordinate system's point of coordinate **v**.

If the coordinate **v** is specified **as incremental value**, the origin of the local coordinate system will be shifted with the offset **v**.
Afterwards, any following motion command specified with absolute coordinates will be executed in this new coordinate system. Positions will also be displayed in this new coordinate system. The values of the coordinates ***v*** will always be handled as ***Cartesian*** data.
The command

G90 G52 v0

cancels the offsets at the point of coordinate v. On power-on, at the end of the program and on reset the offset values set by the command G52 will be cancelled.

If the tool is at the point of the coordinates

X=200, Z=150

in the actual X, Z workpiece coordinate system, the command

```
G90 G52 X80 Z60
```

creates the new X',Z' coordinate system in which the tool will be at the point of the coordinates

X'=120, Z'=90.

The axis components of the offset vector **v'** between the coordinate systems X, Z and X', Z' will be the following by the command G52: v'_x=80, and v'_z=60.

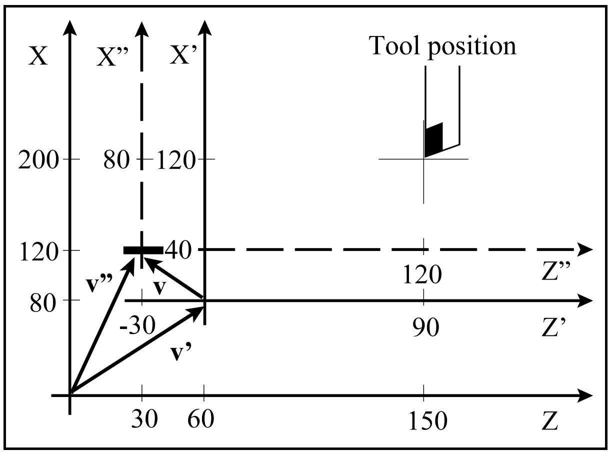


Fig. 10.3-1

A new coordinate system X",Z" can be created in the following two ways.
By ***absolute*** data specification:
The command

```
G90 G52 X120 Z30
```

moves the origin of the coordinate system X",Z" to the point of coordinates X=120, Z=30 ***in the workpiece coordinate system X, Z.*** The components of the vector ***v"*** will be produced by the specification of v''_x=120, v''_z=30.
By ***incremental*** data specification:
The command

```
G91 G52 X40 Z-30
```

shifts the origin of the local coordinate system X',Z' by the values X'=40, Z'=-30. The components of the vector ***v*** will be produced by the specification of v_x=40, v_z=-30. The vector v" showing the position of the new local coordinate system in the workpiece coordinate system X,Z is **v"=v'+v**. Its components are the following: v''_x=80+40=120, v''_z= 60+(-30)=30.
The position of the tool in the coordinate system X",Z": X"=80, Z"=120.

The offset of the local coordinate system is valid in all the workpiece coordinate systems.

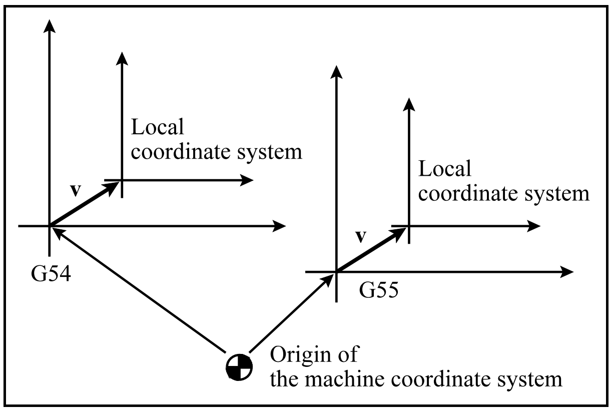


Fig. 10.3-2

Programming the command ***G92*** on the axes for which values were specified, ***cancels*** the offsets created by the command ***G52***, as if the command G52 v0 would have been issued.

If the tool is at the point of coordinates X=240, Z=200 of the workpiece coordinate system X, Z, by the command

```
    G52 X80 Z60
```

its position in the local coordinate system X', Z' will be X'=160, Z'=140. Then, by the command

```
G92 X80 Z110
```

the position of the tool in the new workpiece coordinate system X", Z" will be X"=80, Z"=110. So, the command G92 deletes the local coordinate system X', Z' as if the command G52 X0 Z0 would have been issued.

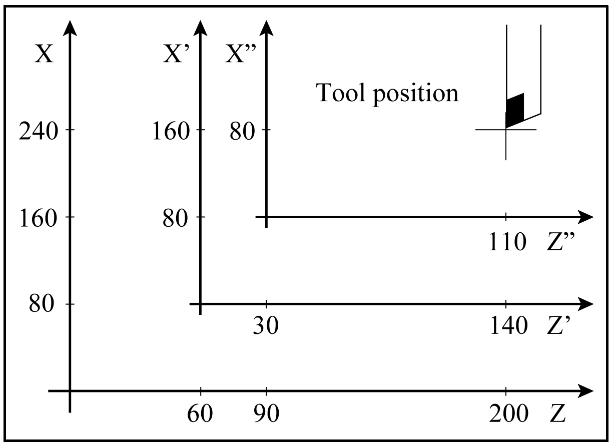


Fig. 10.3-3

10.4 Plane Selection (G17, G18, G19)

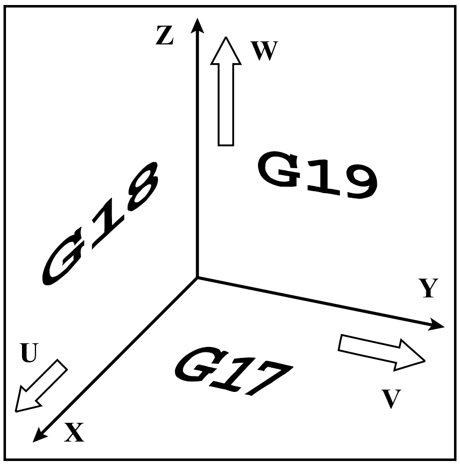


Fig. 10.4-1

The plane in which
– circular interpolation,
– data specification using polar coordinates,
– planar rotation of the coordinate system,
– planar tool radius compensation,
– positionings of drilling cycles,
– turning cycles
are performed, can be selected using the following codes G:

G17	the plane X_pY_p;
G18	the plane Z_pX_p;
G19	the plane Y_pZ_p,

where: X_p: the axis X, or the axis parallel with it;
Y_p: the axis Y, or the axis parallel with it;
Z_p: the axis Z, or the axis parallel with it.
The plane selected is called main plane.
It depends on axis addresses programmed together with the command G17, G18 or G19 in one block, which one of the parallel axes will be selected:
For example, if X and U, Y and V, Z and workpiece are parallel axes, then:
the axis XY will be selected by the command G17 X____ Y____;
the axis XV will be selected by the command G17 X____ V____;
the axis UV will be selected by the command G17 U____ V____;
the axis XW will be selected by the command G18 X____ W____;
the axis YZ will be selected by the command G19 Y____ Z____;
the axis VZ will be selected by the command G19 V____ Z____.
If G17, G18, G19 is not given in a block, the plane selection will remain unchanged:

G17	X____ Y____	the plane XY
	U____ Y____	the plane remain the planeXY.

If axis address is not given in the block G17, G18 and G19, the control will select the main axes:

G17	selects the plane XY;
G17 X	selects the plane XY;
G17 U	selects the plane UY;
G17 V	selects the plane XV;
G18	selects the plane ZX;
G18 W	selects the plane WX.

The motion command does not affect the plane selection; due to the command
```
    G90 G17 G00 Z100
```
the plane XY will be selected and the axis Z moves to the point of coordinate 100.
Changing planes within a program can be done more than once.

After power-on, program end or reset, it will be decided by the bits #1 G18 and #2 G19 of the parameter N1300 DefaultG1 which plane will be valid.

It can be set in the parameter N0103 Axis to Plane which axis address will be used by the control for main and parallel axes.

In this manual there are many references to the first axis and the second axis of the selected plane. The figure below illustrates how to interpret them.

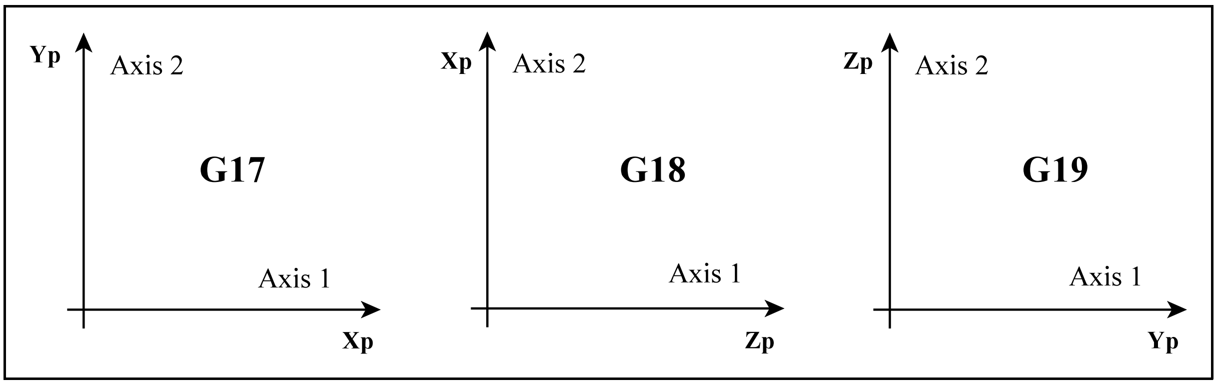


Fig. 10.4-2

11 Spindle Functions

11.1 Spindle Speed Command (Code S)

With writing a maximum 8-digit number at the address

S nnnnnnnn

the code will be transmitted by the control to the PLC.
Depending on the design of the given machine tool, the PLC can interpret the address S either as a code, or as a value with the dimension of revolution/minute.
If motion command and spindle speed (S) are programmed in the same block, the function S will be executed during or after carrying out the motion command. The builder of the machine tool determines the manner of execution.
The speed values specified at the address S are modal ones. After power-on the control starts with the code S0.
In each gear ratio range, the spindle has a minimum and a maximum limit. These limits are determined by the builder of the machine too in the parameter field, and the control does not allow the speed to come out from the range.
The control can manage maximum 8 speed range.

11.1.1 Referring to Several Spindles. Extending the Address S

The control can manage maximum 16 spindles.
If there are several spindles on a machine or in one channel, the address S will not be enough to discriminate the spindles. The control provides two ways for managing several spindles.

Referring to a Spindle at Addresses S and P

The first way is to give the spindle number at the address P in addition to the address S. By the command

Snnnnnnnn Pp

the code S and the spindle number written at the address P will be transmitted by the control to the PLC.
The program language uses the address P for various purposes, for example for waiting, for called subprogram etc. For this reason, the spindle references at the address P must be written in a separate block, otherwise interpretation of the address P will not be unambiguous.

Referring Several Spindles by Extension of the Address S

The another way is to extend the addresses of the spindles. The spindles can also be referred by specifying maximum 3 characters.
The addresses of the spindles must always begin with the letter S. In the parameters N0605 Spindle Name2 and N0606 Spindle Name3, two additional characters may be given which can be letters of the English alphabet: A, B, C, D ... Y, Z or numbers: 0, 1, 2 ... 9. If the spindle name 2 or the spindle name 3 is not used, the value of the parameters will be 0.
Accordingly, the name of a spindle can be SSB, but S1 and S2 may also be used as spindle name.
If the name of a spindle ends with letter, the value belonging to it can be added. The meaning of

```
SSB12500
```

is: the spindle SSB has to rotate at the speed of 12500/min.

If the name of a spindle ends with number, the sign = will have to be written after the name. The meaning of

```
      S1=8700
```

is: the spindle S1has to rotate at the speed of 8700/min.
In the case of referring with an extended spindle address, the speed value programmed on the address and the number of the referred spindle will be transmitted by the control to the PLC.

In the program, ***only one spindle should be referred within one block***. If several spindles have to be started, the commands must be written in separate blocks:

```
      S1=500 M3 (S1 500/min, clockwise)
      S2=1000 M4 (S2 1000/min, counter-clockwise)
```

The control always stores the spindles according to their number. ***Numbering and naming the spindles is also global*** and channel-independent.

11.1.2 Assigning Spindles to Channels

The given ***spindles are always assigned to the given channels by the PLC program***.
Assignment means that speed command for a given spindle can only be issued from the program running in that given channel. For example, if the spindle S4 is assigned to the channel 2, the address S4 cannot be programmed in the channel 1.
During program run, the PLC program can move a given spindle into an other channel due to M function, for example.
Assigning spindles to channels and moving them into other channels is always determined by the builder of the machine tool.
In each channel, a default spindle can be designated in the parameter N0604 Default Spindle. This designated spindle can always be referred at address S even if it has multicharacter name. For example, let the S2 be the spindle No.2. If in the channel 2 N0604 Default Spindle=2, then the spindle can be referred at both addresses S2 and S in the channel 2.

11.2 Functions M Controlling the Spindle

Built-in Functions M

The control manages the spindles using the following built-in codes M:

M3: Spindle On, clockwise
M4: Spindle On, counter-clockwise
M5: Spindle Off (Stop)
M19: Orientation

The direction is always to be meant from the motor towards the spindle. M03, M04 and M05 can also be written instead of M3, M4 and M5.
These are built-in codes M controlling the spindle because they are transmitted by the control for the PLC to stop the spindle, to change its direction of rotation and to orient it during execution of drilling cycles.

Optional Functions M

In addition to the abovementioned built-in codes M, further codes M can also be designated to manage the spindles. These codes can be set in the parameters N0689 Spindle M Low and N0690 Spindle M High, in one array. The smallest value of the array of the codes M has to be written

in the parameter Spindle M Low, but the greatest value has to be written in the parameter Spindle M High.
For example, let Spindle M Low be S2=20, and Spindle M High S2=24. Let the functions of the codes M be the following:

M20: Conversion of the spindle into axis C
M21: Synchronization of the spindle
M22: Synchronization of the spindle together with phase shift
M23: Preparation of the spindle for polygonal turning
M24: Closing the spindle position loop without orientation (Code M for Closing S Loop)

☞ ***Warning!*** *The abovementioned array of the codes M is an example only. The optional functions M of the spindles are always determined by the builder of the machine tool, and for this reason, description of them can be found in the manual of the given machine.*

The optional functions M designated in parameter and the built-in ones are registered by the control as exclusive functions. It means that only one such code M can be written in one block.

The functions M controlling the spindle refer always to the spindle programmed last:

```
S1=1500 M4      (S1 On in M4 direction)
S2=2000 M3      (S2 On in M3 direction)
M5              (S2 Stop)
M19             (S2 Orientation)
S1=0 M5         (S1 Stop)
```

The example above shows, that if the spindle S2 was referred already, the further codes M controlling the spindle will refer to the spindle S2.If, however, the spindle S1should be stopped, it is needed to refer to the address of the spindle.

11.3 Managing the Speed Ranges

There could be a ***variable-ratio drive*** between the driving motor and the spindle which can be used ***for changing the spindle speed range***. Maximum 8 speed ranges for each spindle can be managed by the control. The lower the speed range the spindle is in, the higher the torque will be with which it is able to machining.
For each ratio range, there can be set permissible minimum and maximum speeds, below and above of which the control does not allow the spindle speed to decrease and increase, respectively.
The permissible speeds can overlap each other between the ranges.

The case when speed ranges do not overlap each other

For example:

The minimum speed of the range 1: 50/min
The maximum speed of the range 1: 1000/min
The minimum speed of the range 2: 1001/min
The maximum speed of the range 2: 4000/min

In the case above, according to programming the code S900 it is unambiguous that the spindle has to be rotated in the range 1.
If the ranges do not overlap each other, in addition to the value of ***the address S*** and the number of the referred spindle, the control will transmit the code of the range too to the PLC, and the PLC ***will change*** the required range ***automatically***.

The case when speed ranges overlap each other

For example:

The minimum speed of the range 1: 50/min
The maximum speed of the range 1: 1000/min
The minimum speed of the range 2: 800/min
The maximum speed of the range 2: 4000/min

According to programming the code S900 it is not unambiguous to decide which is the range the spindle has to be rotated in, whether the range 1 or the range 2.
If the ranges overlap each other, the programmer will have ***to use M function to select*** the range in which he would like to rotate the spindle.
These functions M are the following:

M11: Selecting the range 1
M12: Selecting the range 2
...
M18: Selecting the range 8

Managing the ranges on a given machine is determined by the builder of the machine tool, and description of it can be found in the manual of the given machine.

11.4 Main Spindle. Selecting the Main Spindle

If there are several spindles on a machine tool or in a channel, it will have to be decided which one of them will be the 'main' spindle. The following functions apply to the spindle designated as ***main spindle***:

enabling the feed;
feed per revolution;
calculation of constant cutting speed;
threading;
rigid tapping;
master spindle of polygonal turning.

The main spindle has to be designated in each channel. In a given channel, a spindle belonging to an other channel can also be designated as main spindle, for this reason it cannot be programmed in this channel, but the feed per revolution has to be received from this spindle, for example.
The way of selection of the main spindle is determined by the PLC program of the given machine tool, and it is contained ***in the manual provided by the builder of the machine tool***.
Selection can be done using function M, for example:

```
M31 (the spindle 1 acts as main spindle)
M32 (the spindle 2 acts as main spindle)
```

11.5 Controlling the Constant Surface Speed

Function of controlling the constant surface speed can be used for infinitely variable spindle drive only. In this case, the control changes the speed of the spindle in such a way that the tool speed relative to the workpiece surface is always constant and equal to the value programmed.
The control changes always the speed of the spindle designated as main spindle.
The value of the constant surface speed as a function of the input unit system has to be done on the basis of the table below:

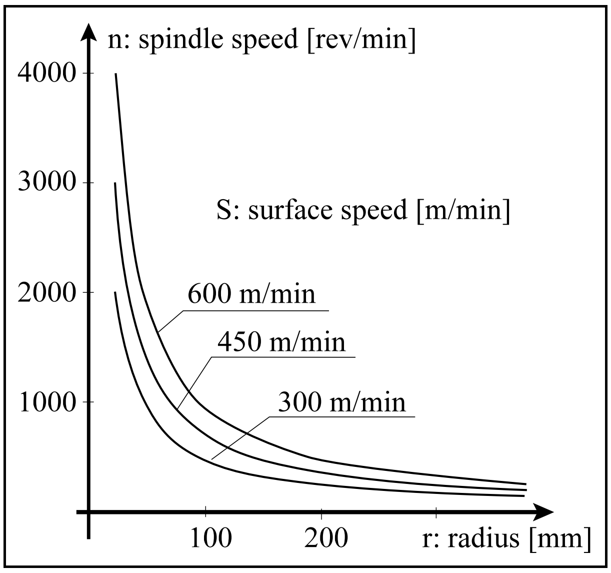


Fig. 11.5-1

Input unit	Unit of the constant surface speed
mm (G21 metric)	m/min
inch (G20 inches)	feet/min

11.5.1 Specifying the Constant Surface Speed Control (G96 S, G97 S)

The command
G96 S
switches the constant surface speed control on. At the address S the value of the constant surface speed control has to be given in the unit shown in the table above.
The value of the constant surface speed control must be given always at the address S, and specification of the speed applies to the spindle designated as main spindle.
For example:

```
M32 (designation of the spindle 2 as main spindle)
G96 S300 (the surface speed is 300 m/min)
```

(M32 here is an example only, the way of designation as main spindle is found in the manual on the given machine tool.)
The command
G97 S
switches the constant surface speed control off .
At the address S, the desired spindle speed can be specified (in rev/min). In the case when there are several axes, after programming G97 always the address of the main spindle has to be used, instead of S. For example:

```
G97 S2=1200 (The speed of the spindle 2 is 1200/min)
```

– For calculation of constant surface speed, the zero point of the axis, on the basis of the position of which the speed of the main spindle has to be changed, must be set on the axis of rotation of the main spindle.
– The constant surface speed control is effective only after starting the main spindle by M3 or M4.
– The value of the constant surface speed is modal even after its calculation has been cancelled by the command G97.

```
G96 S100 M3    (100 m/min or 100 feet/min)
G97 S1500           (1500 rev/min)
G96 X260       (100 m/min or 100 feet/min)
```

– The constant surface speed calculation is effective in mode G94 (feed/min) too.
– If the constant surface speed calculation is cancelled by the command G97 and a new speed of the main spindle is not specified, the last speed of the main spindle taken in the state G96 will remain effective.

```
G96 S100  (100 m/min or 100 feet/min)
     .
     .
     .
G97       (Speed belonging to the resulted diameter X)
```

– In the case of rapid traverse positioning (block G0), the spindle speed is not calculated continuously, but the control sets the speed belonging to the position actual in the endpoint of the positioning.
– After power-on, the parameter N0686 Default Surf Speed determines the value of the constant surface speed.

11.5.2 Clamping the Speed during Calculation of Constant Surface Speed (G92 S)

The command

G92 S

is used for setting the highest speed of the main spindle permissible during constant surface speed control. When the constant surface speed control is switched on, the control disables issuing a main spindle speed greater than the value specified here. In this case, the unit of the S is rev/min. ***The maximum value of the main spindle speed must be given at the address S***, and the speed clamp applies to the spindle designated as main spindle.
– After power-on or if the value of the speed is not clamped by the command G92, in the case of constant surface speed control, the maximum value permissible in the given range will be the upper limit of the main spindle speed.
– In the case of constant surface speed control, a lower limit can also be given for the main spindle speed in the parameter N0688 Min Spindle Speed G96, which can be greater than the minimum value of the speed belonging to the range.
– The value of the maximum speed is modal until a new value is programmed.

11.5.3 Selecting an Axis for Constant Surface Speed Control (G96 P)

In the state G96, the parameter N0687 Default G96 Axis selects the axis, according to the position of which the control calculates the spindle speed.
If an other axis is to be used, the axis according to which the surface speed should be calculated can be specified by the command

G96 P.

The ***address P*** is to be interpreted as ***number of axis***.
In the command G96, address S can also be programmed together with address P:

```
      G96 S300 P4 (surface speed 300 m/min together with the axis
      4)
```

The value set at the address P is modal.

11.6 Spindle Speed Fluctuation Detection

The control monitors speed fluctuation of each spindle. It determines the fluctuation as difference between the programmed speed modified by override and by speed limits, and the actual speed measured from encoder.
If the speed of the spindle is out of the tolerance range set in parameter by the builder of the machine tool, the control will send message to the PLC.
Then, the PLC program sends error message and takes action to stop the spindle and machining. All this is described in the manual provided by the builder of the machine tool.
– The function of speed fluctuation detection will work in the only case if the spindle is equipped with encoder.
– The spindle fluctuation detection is only effective when the spindle rotates (in the state M3 or M4).

11.7 Positioning the Spindles

In case of normal machining, the control issues speed commands proportional to the programmed speed to the spindle drives. At this time, the spindle drive works in the mode of speed control .
In cases of certain technological tasks it could be necessary to bring a spindle into a defined angular position. This process is called spindle positioning or indexing.
Prior to positioning, the control switches the spindle in the mode of position control. Practically, it means that henceforth the control does not issue speed command proportional to the code S, but it measures the spindle position by the use of angular position transmitter (encoder) mounted on the spindle and it issues command depending on desired angular displacement, as it is done on the other position-controlled axes. This is the position feedback.
In order that the spindle on a given machine tool can be positioned, an encoder has to be mounted on the spindle, and the spindle drive should have the design providing operation in the mode of position feedback too.

11.7.1 Spindle Orientation

The function of stopping the spindle in a particular angular position is called spindle orientation or oriented spindle stop. It could be necessary in the case of automatic tool change or for carrying out certain drilling cycles, for example.
The bit state #1 ORI=1of the parameter N0607 Spindle Config means that it is possible to orient a given spindle.
The orientation command is issued by the function

M19.

If there are several spindles on the machine, in addition to the function M19, the spindle also has to be selected. For example:

```
      S2=0 M19
```

Technically, orientation can be done in the following two different ways.

If the spindle cannot be fed back in position control (parameter state of N0607 Spindle Config #2 INX=0), orientation can be carried out by turning on the position switch mounted on the machine.
If the spindle can be fed back in position control (parameter state of N0607 Spindle Config #2 INX=1), the command M19 will cause the control to seek zero pulse of the spindle encoder. Then, the control closes the position control loop automatically.
This, at the same time, means the reference point return of the spindle too, i.e. the spindle can be sent to an absolute angular position after orientation.

11.7.2 Stopping the Spindles and Closing the Position Control Loop

The parameter state N0607 Spindle Config #2 INX=1means that the position control loop can be closed.
In this case, in the parameter N0823 M Code for Closing S Loop there can be given a code M due to which the spindle stops, closes the position control loop, but does not go to the orientation position (does not seek the zero pulse of the encoder).
For example, if the value of the parameter is 24, closing the loop will occur due to the command

```
M24.
```

If there are several spindles on the machine, in addition to the function M, the spindle also has to be selected. For example:

```
S2=0 M24
```

It is the builder of the machine tool who can give information about the code and work of the function.
This function can accelerate execution of rigid tapping cycles, for example.

11.7.3 Programming of the Positioning the Spindles

The parameter state N0607 Spindle Config #2 INX=1means that the position control loop can be closed. This is the only case when positioning the spindles is possible.
The parameter will be set by the builder of the machine tool in the case if the function is implemented on the given spindle.

Positioning on the Basis of Axis Name

Each spindle may have a maximum 3-character-long axis name which can be referred to after closing the position control loop.
In the parameter N0817 Spindle Axis Name1, it is obligatory to set the letter A, B or C. In the second character (parameter N0818 Spindle Axis Name2) and on the third character (parameter N0819 Spindle Axis Name3) there can be given letters A, B, C, G ...Y, Z of the English alphabet or numbers 0, 1, 2 ... ,9.
For example:

CS: the name of the spindle axis 1
CS2: the name of the spindle axis 2
ABC: the name of the spindle axis 3

If the name ends with number, the sign = will have to be used. Certainly, the name given should not coincide with other names.
After the address, ***position has to be given in degree***. Positioning is executed at the rapid traverse speed set for the spindle axis. The unit of the rapid traverse is 1/min.

Prior to the ***positioning*** by ***absolute*** data specification, M19 has to be programmed. For example: The command lines

```
        S2=0 M19
        G90 CS2=30
```

moves the spindle CS2 to the position of 30 degree, rotating the spindle in positive direction.
In the case of absolute positioning, the control ***cuts the full turns*** from the data greater than 360 degree, for example, if

```
        S2=0 M19
        G90 CS2=750
```

are given, the control will rotate the spindle to the 30 degree. The control rotates ***always in the direction of the shorter travel***. If

```
        S2=0 M19
        G90 CS2=270
```

are given, the control will rotate to the 270 degree, in negative direction.

Prior to the ***positioning*** by ***incremental*** data specification, M19 has not to be programmed, it is enough to close the position loop only. If the loop is closed by the M24, then:
in the case of the following data specifications

```
        S2=0 M24
        G91 CS2=3600
```

or

```
        S2=0 M24
        CS2=I3600,
```

the spindle will make 10 revolutions in the given direction (here in positive direction). In the case of incremental data specification, the control ***does not cut the full turns and will rotate in the direction given by the sign***.

Indexing the Spindles using Function M

Spindles can be indexed using codes M. Indexing means revolving the spindle to a discrete preset positions.
For this, the following parameters have to be set:
The direction of indexing at the bit #7 IDS of the parameter N0607 Spindle Config: =0 positive, =1 negative.
The initial value of the codes M in the parameter N0820 Start M of Spnd. Pos. and the the number of the codes M in the parameter N0821 No. of M Code for Spnd. Pos..
The angle of indexing in the parameter N0822 Basic Angle of Spnd. Pos..

An example:
The spindle can be fixed by 18 degrees starting from the orientation position. The setting is as follows:
Spindle Config: #7 IDS=0: indexing in positive direction
Start M of Spnd. Pos.=201 (m=201)
No. of M Code for Spnd. Pos.=20 (n=20)
Basic Angle of Spnd. Pos.=18 (φ=18)

The following table explains the different codes M:

Code M	The revolved angle α
Mm (M201)	α=φ=18°
M(m+1) (M202)	α=2φ=36°
M(m+2) (M203)	α=3φ=54°
....	
M(m+n) (M220)	α=nφ=360°

On the basis of the settings above, let's have the example below:

```
    M19      (orientation)
    M205     (revolving by 90°)
    ...      (drilling)
    M210     (revolving by 180°)
    ...      (drilling)
```

Before issuing a code M for positioning, the spindle should be oriented by M19. Thus the spindle gets the position α=0.
A hole has to be drilled at the position 90°, the spindle revolves to the position α=5φ=5*18=90° by the command M205.
The next hole has to be drilled at the position 270°. When code M is used, the spindle can be moved only incrementally, for this reason M210 has to be programmed, because the displacement will be 10*18°=180° in this case.

11.7.4 Position-correct Synchronization of Two Spindles

Synchronization of two spindles means their rotation at the same speed and their being in a preset phase shift to each other during rotation.
Position-correct synchronization of two spindles is done by the PLC program, using generally a code M. The method of synchronization is contained in the documentation provided by the builder of the machine tool.
For synchronization of two spindles, it is required of both spindles to be equipped with encoder, and that the position control looping can be realized.
During synchronization, ***the maximum speed of spindles*** is limited by ***the rapid traverse speed*** set for the spindle axis. Generally, this speed is lower than the maximum spindle speed.
In the course of synchronization, there are differentiated master and slave spindles. Always the slave spindle is synchronized to the master one.

The process of synchronization is as follows:
– if the speed of the master spindle is higher than the lower one of the rapid traverse speeds specified for the two (master and slave) spindles, the master one will slow down to the proper speed;
– the master spindle closes the position control loop;
– the slave spindle accelerates up to the master spindle speed in the same or opposite direction of the master spindle rotation;
– the slave spindle closes the position control loop;

– then, the slave spindle pulls its zero pulse over the zero pulse of the master spindle, or it executes a phase shift from the zero pulse of the master spindle on a distance given in encoder pulse and specified in the parameter N0685 Spindle Phase Shift.

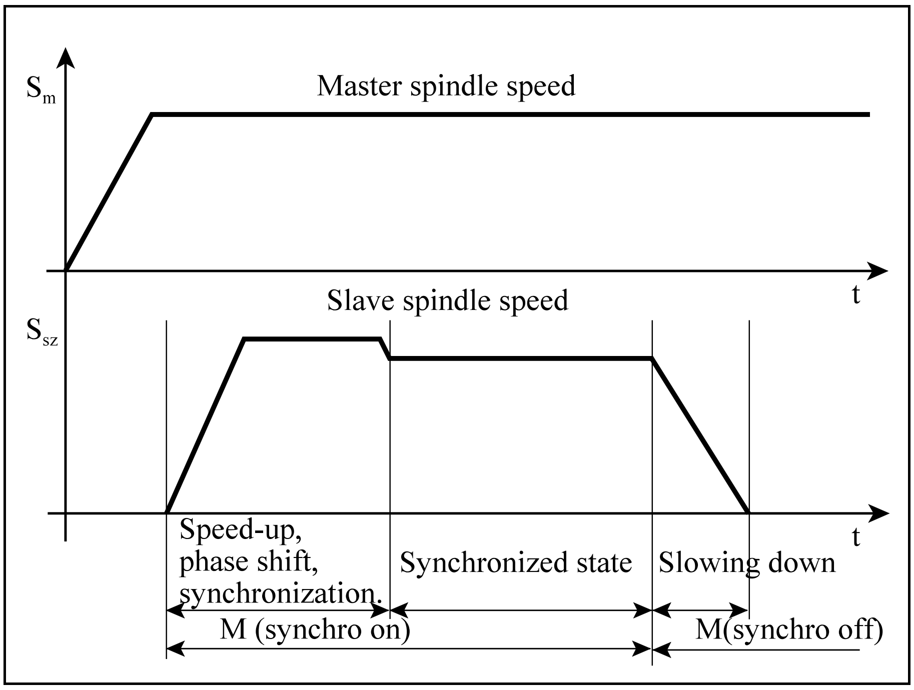


Fig. 11.7.4-1

An example:
It is required to machine the another side of a workpiece cut in the main spindle (marked as S1) of a sub-spindle lathe. The material the main spindle uses is bar. First, the sub-spindle being the slave spindle (marked as S2) should be synchronized to the main (master) spindle, the slave spindle should clamp the workpiece, and then cutoff should be executed.
Let M21 be synchronization realized by synchronous run of the zero pulses and let M22 be pulse shift synchronization.
If pulse shift is not necessary, the program detail will be the following:

```
S1=3000 M3
S2=0 M21  (synchronizing the S2 to S1)
...       (clamping the workpiece by the chuck of the S2)
...       (cutting off)
S2=1200 M4(turning the synchronization off, rotating the S2)
```

If a shaped workpiece should be clamped, the code M22 will have to be used and phase shift will have to be set in the parameter N0685 Spindle Phase Shift.

☞ ***Warning!*** *The example above is a sample only. In a particular case, the process is as it is directed by the builder of the machine tool.*

11.7.5 Turning the Position-controlled Operating Mode off

After orientation, positioning or synchronization, the following function must be used for turning the spindles off from the position-controlled operating mode:

M3, M4 or M5

For example:
```
     S1=0 M19  (position-controlled mode on, orientating the S1)
     CS1=60
     ...
     S1=0 M5   (position-controlled mode off, spindle stands)
```
or
```
     S1=2400 (position-controlled mode off, spindle on)
```

11.8 Converting Spindle into Axis and Axis into Spindle

Spindle axes can be used for machining with limitations only because they can be positioned only, and they cannot participate in interpolation with other axes. Furthermore, spindle axes use low-resolution and high-speed encoder, while a rotary table requires high-resolution encoder operating at low speed.
Therefore, in the course of machining it could be necessary to convert a spindle into axis or an axis into spindle. The spindle of a lathe has to be converted into axis C in order to mill on the face of the workpiece using polar coordinate interpolation or to engrave something onto its curved surface using cylindrical interpolation. Then, the axis C has to be reverted to spindle in order to execute further turning operations.
If the construction of the machine makes it possible, the rotary table B of a horizontal machining center can be converted into spindle in order to execute turning operation on the workpiece.

Appropriate mechanical and electronic construction of the machine tool is necessary to convert a spindle into axis or an axis into spindle. The manual of the given machine contains information about whether such conversion is possible on the machine. Always the PLC executes conversion in accordance with capabilities of the given machine.
Usually, the process of conversion of ***a spindle into axis*** is as follows:
– stopping the spindle if it rotates;
– stopping drive operation;
– disconnecting the spindle from the drive;
– converting the encoder into a high-resolution one;
– adjusting the drive parameters;
– turning the drive on again;
– connecting the axis to the drive and makes its display visible.
Usually, the process of conversion of ***an axis into spindle*** is as follows:
– waiting until the axis stops;
– stopping drive operation;
– disconnecting the axis from the drive and makes its display invisible;
– converting the encoder into a low-resolution one;
– adjusting the drive parameters;
– turning the drive on again;
– connecting the spindle to the drive.

An example:
On a lathe, the spindle S1 should be converted into the axis C1 for milling, and then, the axis C1 shoul be reverted to the spindle S1for further turning:
```
     S1=0 M20       (converting S1 into C1)
     G28 G91 C1=0   (reference point return on the axis C1)
     ...                 (milling by the use of the axis C1)
     S1=3000 M3     (reverting C1 to S1, rotation by 3000)
     ...            (turning by the use of S1)
```

☞ ***Warning!*** *The example above is a sample only. In a particular case, the process is as it is directed by the builder of the machine tool.*

12 Function T

In the part program, the tool number is referred by ***code T***. The code T should be given with a maximum ***8 decimal digit***:

Tnnnnnnnn

The leading zeros can be eliminated.

12.1 Programming the Tool Change

There are basically two ways of referring to tool change in the part program. These two ways depend on the machine construction. The tool call technique applicable in the part program is defined by the builder of the machine tool.

Case A: Tool change on code T

Tool change on the machine can be carried out manually or using turret-type tool changer.
At the bit position #0 TCM=0 of the parameter N1414 Comp. Config on Lathes, ***the code T includes the code of tool compensation too*** at the digits of low place value. The tool number can be given at the remaining digits of great place value. At the bit position TCM=0, tool change occurs too when tool number is called.
It is the parameter N1413 No. of Digits of Offs. No. in T Code which determines how many digits are needed to specify the tool compensation. The value of the parameter can be 0, 1, 2 and 3.

Tnnnnnnnk: compensation on one digit
Tnnnnnnkk: compensation on two digits
Tnnnnnkkk: compensation on three digits

where:

nnnnnn: the number of the tool
kk: the number of the compensation location

If compensation is given on two digits, the meaning of the command

```
T1236
```

will be:

the tool of number 12 is to be changed, and
the compensation group of number 36 is to be called too.

If 0 is programmed for the tool number or only so many digits are programmed at the address T as many digits are needed for specifying the compensation, tool change will not occur but new compensation will be called. Using the setting above, the meaning of

```
T12
```

is for example: the compensation group of number 12 is to be called and the previous tool remains changed.

If motion command and tool number (T) is programmed in the same block, the function T will be executed during or after execution of the motion command. The builder of the machine determines the manner of execution.
The tool number will be given to the PLC program.

Case B: Tool change on the function M6

If tool preparation on the machine is needed for tool change, i.e. the tools are in a magazine and they can be changed by arm-type changer, the tool number will not have to include the compensation code, because the code T prepares the tool for change only, and the change will occur later due to the function M6.

In this case, at the bit position #0 TCM=1 of the parameter N1414 Comp. Config on Lathes ***the code T does not include the code of tool compensation***.

Tool preparation on the machine is needed for tool change. The steps of preparation are as follows:

– Finding the tool to be changed in the tool magazine. In this step, referring to the address

  Tnnnnnnnn

  in the part program triggers motion of the appropriate tool to the change position. This action goes on in the background, parallelly with the machining.

– Sending the slides to the change position.

– Executing the tool change by the function

  M6

  in the program. (M06 can also be used.) The control waits for executing the tool change until the tool T being under preparation arrives at the change position. Then it inserts the new tool into the tool holder. From this point, machining can be continued.

– Inserting the previous tool back into the tool magazine. This action goes on in the background, parallelly with the machining.

– Beginning to find the new tool in the tool magazine.

In the case of tool change on the function M6, the following parameter settings are required:

Bit position #0 M06=1 of the parameter N1338 Block No Search - for block search

Bit position #1 TLC=0 of the parameter N2901 Search Config - for tool life management

An example:

```
T12 M6    (changing the tool T12)
T15       (calling the tool T15, the PLC searches in the
          course of machining)
...       (machining by the use of the tool T12)
M6        (changing the tool T15)
T8        (calling the tool T8, the PLC searches in the
          course of machining)
...       (machining by the use of the tool T15)
T9        (calling the tool T9, the PLC searches in the
          course of machining)
M6        (changing the tool T8)
...       (machining by the use of the tool T8)
```

If command T and command M6 are written in one block, the PLC will usually execute function T at first, and then function M6, i.e. it will execute tool change. After tool change, M6, it is practical to program tool call in the next block in order to minimize secondary time of the machine.

☞ ***Warning!*** *The example above is a sample only. In a particular case, the process is as it is directed by the builder of the machine tool.*

13 Miscellaneous and Auxiliary Functions

13.1 Miscellaneous Functions (Codes M)

Having written a numerical value of maximum 8 digits behind the address **M**
Mnnnnnnnn,
the NC transfers the code to the PLC.
The leading zeros can be eliminated.
Eight several codes M can be transferred by the control to the PLC at the same time, i.e. ***maximum 8 codes M can be written in one block***.
The ***execute sequence*** of the functions M written in one block is determined ***in the PLC program*** by the builder of the machine tool.
If motion command and miscellaneous function (code M) are programmed in the same block, the miscellaneous function will be executed during or after execution of the motion command.
All the codes M, even those that are executed by the control, will be transferred by the control to the PLC.
The way of execution is determined by the builder of the machine tool.

The program control codes M:
M0: programmed stop
This code is executed by the PLC. At the end of the block in which the M0 is specified, the control
arrives at the condition Stop;
stops the spindles;
cuts off the coolant.
All the modal functions remain unchanged. Due to the start, the control restarts the spindles, switches back the coolant and continues the program.

M1: conditional stop
This code is executed by the PLC. Its effect is the same as the effect of the code M0. It will only be executed when the button CONDITIONAL STOP is activated. If the appropriate button is not activated, this code will be ineffective.

M2, M30: end of program
This code is executed by the PLC. It means the end of the main program. The machine functions are reset by the PLC program; generally, it stops the rotation of the spindles and switches off the coolant.
Each of the commands M2 or M30 executed increases the value of the workpiece counters by one, unless other code M is assigned in the parameter N2305 Part Count M for stepping the the counter.

M96: enabling the interruption macro
This code is transferred to the PLC but it is executed by the control. It enables the interruption signal coming from the PLC, due to which the interruption macro will be called.

M97: disabling the interruption macro
This code is transferred to the PLC but it is executed by the control. It disables interruption signal coming from the PLC to be valid, and running the interruption macro.

M98: calling subprogram
This code is transferred to the PLC but it is executed by the control. Due to it, calling a subprogram will be occurred.

M99: end of subprogram
This code is transferred to the PLC but it is executed by the control. Due to it, execution returns to the position of the call.

The spindle control codes M
M3, M4, M5, M19: the codes of spindle management
These codes are executed by the PLC. See the sub-chapter ***Functions M Controlling the Spindle***.

The spindle management codes M that can be specified in parameter
In the parameters N0689 Spindle M Low and N0690 Spindle M High, additional ***spindle control codes M*** can be assigned in one array. See the sub-chapter *Functions M Controlling the Spindle*.
The ***spindle control codes M*** together with the codes ***M3, M4, M5, M19*** are ***exclusive ones***, giving only one such code in one block is allowed.
These codes are executed by the PLC.
The initial value and the number of the ***codes M indexing the spindles*** can be specified in the parameters N0820 Start M of Spnd. Pos. and N0821 No. of M Code for Spnd. Pos., respectively. See the sub-chapter *Programming of the Positioning the Spindles*.
These codes are executed by the control.

Speed ranges management codes M
M11, ..., M18: the codes of speed range change
These codes are executed by the PLC.

Tool change code M
M6: the code of the tool change
This code is executed by the PLC.

Code M group that can be set in parameter
There can be assigned ***16 different groups of code M*** in 16 pairs of parameter. These are executed by the PLC.
The code of the lowest number in the group has to be written in the parameters N1341 M GR Low 1, ..., N1356 M GR Low 16; the code of the highest number in the group has to be written in the parameters N1357 M GR High 1, ..., N1372 M GR High 16.
The ***groups of codes M*** must be specified in such way so that the codes mean ***exclusive machine statuses***.
During execution of the program, the control ***filters the codes M in such a way*** that only one of the Codes M in the group is in the given block, otherwise the control sends error message *Conflicting M codes*.

When it gathers the codes M, the control takes the values preset in the parameters into account in the course of ***finding blocks***, too. From the codes M related to a group, the only code given last will be gathered by the control.
The values of these codes M will be displayed by the PLC program on the screen in the ***window of codes M***, too.
For example:
Let the codes M of opening and closing the chuck be the following:
M51: opening the chuck
M52: closing chuck
M53: opening the chuck when the spindle id rotating
The parameters are set as follows:
N1341 M GR Low 1=51
N1357 M GR High 1=53
The codes M of chuck gripping modes:
M54: gripping from the outside
M55: gripping from the inside
The parameters are set as follows:
N1342 M GR Low 2=54
N1358 M GR High 2=55
In the part program, writing either M51 or M52 or M53 is allowed in one block, otherwise error will be indicated by the control during program run. This is related to the group of M54 and M55, too.
During block finding, from among the M51, M52 and M53, the control gathers and gets it executed the only one programmed last. This is related to the group of M54 and M55, too.
From among the appropriate machine statuses, the only status code valid in the group will be displayed in the window of codes M.

The codes M carrying out synchronization of the channels
In the parameters N2201 Waiting M Codes Min and N2202 Waiting M Codes Max, there can be assigned a gruop of codes M, using which synchronization, waiting for each other can be realized. It is executed by the control.

13.2 Auxiliary Functions (A, B, C, U, V or W)

In addition to the addresses M, S and T, there can be assigned in parameter further 3 addresses at which auxiliary function can be transferred to the PLC program. All the 3 auxiliary functions can be transferred by the control at the same time.
In the parameters N1333 Aux Fu Addr1, N1334 Aux Fu Addr2, N1335 Aux Fu Addr3, from among the addresses A, B, C, U, V and W there can be selected ones at which auxiliary functions can be transferred
For the auxiliary functions, value can be given using number of maximum 8 decimal digits
If motion command and auxiliary function (code M) are programmed in the same block, the auxiliary function will be executed during or after execution of the motion command.
The execute sequence is determined by the builder of the machine tool, and it is contained in the machine tool specification.
For example, at the address B, indexing of the indexing table can be realize.

13.3 Buffer Emptying Functions

The block processor in the control ***reads ahead, processes the blocks, and then buffers them***. The executive element (the interpolator and the PLC) takes the processed blocks from the buffer, and then executes motions and functions specified in the blocks
In certain cases, it could be necessary ***to stop reading ahead*** the blocks ***in order to synchronize action between the control and the PLC***.
For example, if the PLC asks the NC for one or more axes in order to move them, reading ahead will have to be suspended. Reading ahead can be continued after execution of the function only, when the PLC gave back the axes to the NC. Then, the NC can continue the machining from the axis position changed by the PLC.

Ten single codes M, eight groups of codes M, all the three auxiliary functions and the codes S and T can be assigned in parameter for buffer emptying.
The buffer emptying functions are determined and set by the builder of the machine tool.

For example, in the course of execution of the program, reading ahead the blocks is necessary to take the tool radius compensation into account (G41 and G42). If buffer emptying function is programmed during G41and G42, the control suspends calculation of the tool radius compensation, and as a result of this, the contour will damage.

14 Part Program Configuration

The structure and the format of the part program have already been shown in the introduction. In this chapter, organizing the part programs will be dealt with.

14.1 Block Number (Address N)

The blocks of the program can have ***serial number***. The block numbers can be managed as ***labels*** which can be referred to in the other parts of the program. Blocks are numbered by the command

Nnnnnnnnn.

Maximum 8 digits can be written at the address N. It is not obligatory to use address N. Some blocks could be numbered, others not. The block numbers need not follow each other in a consecutive order.

14.2 Conditional Block Skip (/ address)

Conditional block skip can be programmed using slash command

/n.

The ***value of the slash address can be n=1-8***. The numbers from 1 up to 8 are serial numbers of switches. The conditional block switch No.1 can be found on the operator's panel of the control. Mounting the other switches is optional and is determined by the builder of the machine tool.
If a conditional block skip /n is programmed at the beginning of a block,
– that block ***will be omitted*** from the execution when the nth switch is ***on***,
– that block ***will be executed*** when the nth switch is ***off.***

If address /only is programmed at the beginning of the block, that will be related to the switch 1:

```
/ N1200 G0 X200    (the block will be omitted if the
                   switch 1 is on)
```

It can be programmed in the following way, too:

```
/1 N1200 G0 X200   (the block will be omitted if the
                   switch 1 is on)
```

If the intention is that a switch of conditional block skip should be taken into account by the control even in the block preceding the execution of the conditional block, the parameter N1337 Execution Config will have to be set in ***#4 CBB=0***. Then, the command of conditional block (blocks beginning with the character /) ***suppresses*** the reading ahead of block. In this case, using ***G41 and G42***, the contour ***becomes distorted***, but it is enough to ***turn the switch of conditional block skip on during execution of the previous block*** in order that it will be effective.

In order that the command of the character / will not suppress the reading ahead of block, the parameter N1337 Execution Config will have to be set in ***#4 CBB=1.*** Then, the command of conditional block (blocks beginning with the character /) ***does not suppress*** the reading ahead of block. In this case, using ***G41 and G42***, the contour ***does not become distorted***, but the switch of conditional block skip ***has to be turned on before execution of the program***, for sure effectiveness.

Some switches can also be used by the PLC programin order to control the program run.

For example, in the case of a machine equipped with workpiece feeder, the main program can be made endless using M99:

```
... (part program)
M90 (stepping the workpiece counter)
/8 M30 (the switch 8 is controlled by the workpiece counter)
M99
```

When the workpiece counter reaches the needed workpiece quantity, the PLC turns the switch 8 of conditional block skip off, the program runs to the M30, and the execution stops.
The example above is correct when the parameter N1337 Execution Config is in the state #4 CBB=0, i.e. the switches / suppress the reading ahead of block.
When the parameter N1337 Execution Config is in the state #4 CBB=1, the program will run correctly in that case only if the code M90 is set for buffer emptying.
Before changing the parameter, please ask the builder of the machine tool for information about effects!

14.3 Writing Comments into the Part Program: (comment)

If a part of the program is parenthesized between ***round brackets ( )***, the section between the brackets will not be taken into account by the block processor.
Thus, notes (comments) can be written into the part programs.
If the intention is that a part of the part program will not be executed by the controller but that program part will not be deleted from the program, that given part will have to be put into round brackets.

For example:

```
N10 G0 X100    (positioning to X100)
N20 Z30
(N30 G1 Z60 F0.3
N40 X300)
...
```

A comment is written in the block N10. The block N30N40 is put into brackets, so these two blocks will be taken into account by the control.

14.4 Main Program and Subprogram

Two kinds of program are distinguished: main program and subprogram. Macro is a subprogram to which arguments can be transferred.
In the course of machining a part, repetitive actions could be encountered, which can be described by the same program part. In order to avoid writing the repetitive parts many times in the program, from these parts subprogram can be organized that can be called from the main program.
Structure of a main program and a subprogram is given in the Introduction.
The difference between them is as follows: after execution of a main program, machining is completed and the control waits for restart; but after execution of a subprogram, the execution returns to the calling program and continues machining from there.
In terms of programming technique, the difference between the two programs lies in the way of terminating the program. The end of the main program is indicated with the codes M02 or M30 (it is not obligatory to use them), whereas the subprogram must be terminated with the code M99.

14.4.1 Identification of Programs in Memory. The Program Number (O)

In the memory, programs are placed in folders defined by the user and having different names. Programs in the folders are identified by their file name. The control will consider a file as a part program, i.e. a file can be run as a part program if its extension (the fraction after the dot '.') is as follows:

    .txt
    filename.prg
    filename.nct
or
    filename.nc

The file name of a program can be composed of alphabetical and numerical characters.
Subprograms are stored in separate file, they together with the main program have not to be in one file.

Program number

The program number

Onnnn.ext

or

Onnnnnnnn.ext

is a special filename beginning with the letter O obligatorily, and followed by 4 or 8 decimal digits. For their extension (.ext), the notes above are valid.
Onnnn: ***Letter O followed by 4 digits*** together with the leading zeros.

O1 is an invalid filename,
O0001 is a valid filename;

or

Onnnnnnnn: ***Letter O followed by 8 digits*** together with the leading zeros.

O01234 is an invalid filename,
O00001234 is a valid filename.

14.4.2 Calling a Subprogram (M98)

Subprograms can be called in two ways, by program number or by filename.

Calling a subprogram by program number

The command line

M98 P....

generates a subprogram call. The program number of the called program is written at the address P. At the address P, the leading zeros can be eliminated and it is not allowed to write extension after the number. Due to the command, execution of the program will continue at the subprogram the number of which defined at the address P:

```
calling program                    subprogram         note

O0010                                                 execution of the
......                                                program O0010
......
M98 P11          --->              O0011              calling the
                                                      subprogram O0011
```

```
                                   ......            execution   of   the
                                   ......            subprogram O0011
                                   ......
next block  <---                   M99               return to the calling
                                                     program
......                                               continuing    the
......                                               program O0010
```

In the case of subprograms called at the address P and by program number, the following limitations concerning their location in the folder system and their filenames are valid:

– The subprograms have to be in the same folder where there is the program calling them.

– For filename of the subprograms called by program number, the limitations described in the previous subchapter are valid.

– ***The extension of subprograms and the extension of the program calling them have to be the same***:

For example, if the filename of the main program is

Foprogram.**prg**,

the extension of the subprogram called from the 'Foprogram.prg' will have to be .prg, too:

in the case of O1234.**prg**	the call is executed;
in the case of O1234.**nct**	error message is generated.

Calling a subprogram by filename

The command line

M98 <alprogram.nct>

calls the subprogram named 'alprogram.nct' being together with the program in the same folder. The ***filename*** has to be written in between the symbols **<** ***less than*** and **>** ***greater than***.
In this case the extensions of the main program and the subprogram do not need to be the same. ***The relative path of the file can also be given between the symbols < and >.*** The path has to be always given from the folder of the calling program.

If the called subprogram 'subprogram.prg' is in a subfolder named 'subprograms' being ***a level lower*** from the folder of the program 'program1.nct' which calls it:

...
subprograms (folder)
program1.nct (file)

the subprogram call will be executed by the command

M98 <\subprograms\subprogram.prg>.

If the called subprogram 'subprogram.prg' is in ***a level upper*** from the folder 'mainprograms' of the program which calls it:

...
mainprograms (folder)
subprogram.prg (file)

the subprogram call will be executed by the command

M98 < ..\subprogram.prg>.

Steppings backward and forward can be done through several levels.
For example:

< ..\..\..\folder1\folder2\folder3\file.txt>

However, the ***length of the text between the symbols < and >*** can be ***maximum 60 characters.***

☞ ***Note:*** *In the course of giving the path, the symbol \ (backslash) has to always be used; it should not be confused with the symbol / (per).*
For stepping backward in the folder system, 2 dots (..) has to always be used.

Calling a subprogram by number of repetitions

The command line

M98 P.... L....

or

M98 <path \ filename> L....

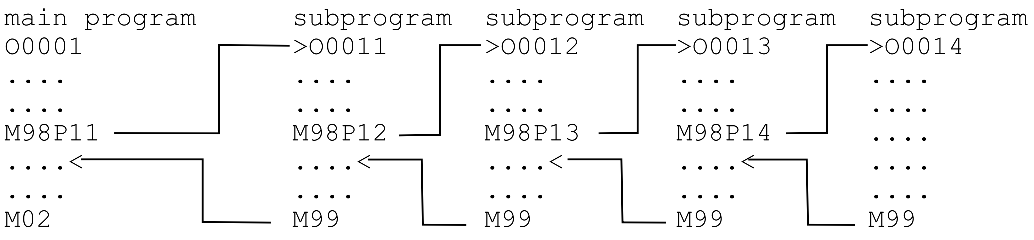
calls in succession the specified subprogram so many times as many times given at the address L.
The address L can be specified by maximum 8 decimal digits.
If value is not given to the L, the subprogram will be called once, i.e. L=1 is assumed by the control.
The command

```
M98 P11 L6
```

means that the subprogram O0011 has to be called 6 times in succession.

Multi-level subprogram call

A subprogram can be called from another subprogram, too. Subprogram calls (together with macro calls) can be nested to maximum ***16 levels***.



14.4.3 Return from a Subprogram (M99)

Return to the block following the call

Using the command

M99

in a subprogram means the end of that subprogram, and control will be given back to that block of the calling program which follows the call:

calling program		**subprogram**	**note**
O0010			execution of the program O0010
N101 M98 P11	--->	O0011	calling the subprogram O0011
		....	execution of the subprogram O0011

```
N102 ......            <---      M99              return  to  the next
                                                  block of the calling
                                                  program
......                                            continuing    the
......                                            program O0010
```

Return to a given block

Using the command

M99 P...

in a subprogram means the end of that subprogram, and control will be given back to that block of the calling program the number of which was specified at the address P. Maximum 8 decimal digits can be written at the address P.

```
calling program                  subprogram       note

O0010                                             execution   of   the
......                                            program O0010
......
......
N101 M98 P11      --->           O0011            calling   the
                                                  subprogram O0011
                                 ......           execution   of   the
                                 ......           subprogram O0011
                                 ......
N250 ......            <---      M99 P250         return  to the block
                                                  N250  of  the calling
                                                  program
......                                            continuing    the
......                                            program O0010
```

Return by modification of cycle counter

The command

M99 (P...) L...

modifies the cycle counter of the calling program. If 0 is written at the address L, the subprogram will be called once only. For example, if the subprogram O0011 is called by the command

```
M98 P11 L20,
```

and return from there is executed by the command

```
M99 L5,
```

the subprogram O0011 will be called 6 times altogether.
Maximum 8 decimal digits can be written at the address L.

14.4.4 Jump within the Main Program

Using the command

M99

in the main program produces an unconditional jump to the first block of the main program, and the control continues execution of the program from here. Using the command results in endless cycle:

```
O0123
N1... <
...
.....
.....
M99
```

Using the command

M99 P.....

in the main program produces an unconditional jump to that block of the main program the number of which is specified at the address P, and the control continues execution of the program from here. Using the command results in endless cycle:

```
O0011
....
....
N128....<
....
....
M99 P128
```

```
O0011
....
M99 P225
....
....
N225   <
....
```

The program can be recovered from the endless cycle either by reset, or by programming the block containing the command M99 with conditional block skip

```
/ M99;
```

depending on the position of the conditional block skip switch, jump will happen or not.

14.5 Functions M of Channel Synchronization

In the course of multi-channel operation, it could be necessary that the running of a program in a channel has to wait at a given point until a program or programs running in one or more other channels come to execution of certain operations. This is the synchronization among channels. This synchronization can be realized by the use of so-called codes M for waiting .

The codes M for waiting are ***program organizing codes M***; ***they are processed by the control*** and they are not transferred to the PLC. ***They are buffer emptying codes M***, i.e. preprocessing the blocks is being intermitted until synchronization is completed in all the channels, and it will only be continued after synchronization.

Maximum 100 codes M for waiting can be assigned in parameter. The initial value of the group can be given in the parameter N2201 Waiting M Codes Min, and the final value of the can be given in the parameter N2202 Waiting M Codes Max.

If, for example,

N2201 Waiting M Codes Min=500 and

N2202 Waiting M Codes Max=599,

then the codes M M500, M501, M502, ... , M599 M can be used for waiting.

The command

Mm Pppppppp

will program the waiting among two or more channels. The waiting has to be programmed in separate line.

m: one of the codes M for waiting specified in parameter,

pppppppp: number of those channels among which the waiting has to be done. Since there can be maximum 8 channels in the system, the address can have maximum 8 digits.

If, for example, the waiting has to be done between the channels 1 and 2, the command

```
M501 P12
```

will have to be written into both programs, to the appropriate point.

The PLC can ignore the waiting by function M or by push-button, using the flag CP_NOWT. It can be useful in the case, when the program should be run in one of the channels only and the codes M for waiting should not be glossed.

An example:

Let the minimum and maximum values of the codes M be 500 and 599, respectively; and let there be 3 channels:

The program of the channel 1	**The program of the channel 2**	**The program of the channel 3**
... machining	... machining	... machining
N60 M501 P12		
waits for the channel 2	N100 M501 P12	
... machining	...machining	N110 M502 P123 waits for the channels 1 and 2
	M502 P123 waits for the channels 1 and 3	
N130 M502 P123		
... machining		
	... machining	... machining

Explanation:

First, the channel 1 runs to the code M501 and waits until the channel 2 runs to it, too. After synchronization, both channels continue its own program. Meanwhile, the channel 3 operates continuously.

It is the channel 3 that runs to the code M502 first of all, then the channel 2 does this, and then the channel 1, last of all. The channel 3 has to wait until the channels 1 and 2 run to the code. After reaching this point, machining in all three channels can start.

15 Tool Compensation

In order that overhang and radius etc. values related to different tools should not be taken into account in the part program in the course of specifying the coordinates, tool characteristics are gathered in a so-called offset table. Whenever a tool has to be called in the part program, it has to be specified where the characteristics of that given tool can be found in the offset table. According to this, the control directs the tool along the programmed path, taking the referred offsets into account.

15.1 The Compensation Memory. Referring to Tool Compensation (T or D)

The address at which the tool compensation can be referred to is determined by the mechanism of the tool changer. See the chapter The Function T.

The case A

In the case of turret-type tool changer, at the bit state #0 TCM=0 of the parameter N1414 Comp. Config on Lathes ***the code T contains the code of tool compensation too*** at the digits of low place value. The tool number can be given at the remaining digits of great place value.
It is the parameter N1413 No. of Digits of Offs. No. in T Code which determines how many digits are needed to specify the tool compensation. The value of the parameter can be 0, 1, 2 and 3.

Tnnnnnnnk: compensation on one digit
Tnnnnnnkk: compensation on two digits
Tnnnnnkkk: compensation on three digits

where:

n: the number of the tool
k: the number of the compensation

The case B

If the tool change is executed by the code M6, at the bit state #0 TCM=1 of the parameter N1414 Comp. Config on Lathes ***the code T does not contain the code of tool compensation.*** In this case, the compensation cell, the length and radius compensations together can be referred to at the address D:
to the length and radius compensations: at the address **D**
The number following the address is the compensation number, and it indicates the compensation value to be called. The range of the address D is 0-999. The leading zeros can be eliminated.

Distribution of the compensation memory among the channels

The ***quantity of the tool compensation groups*** to be accessible in a given channel can be given ***for each channel*** in the parameter N1400 No. of Tool Offsets. Length and radius compensations also pertain to each group.
In the whole system, ***the sum of the lathe channel compensation groups*** has not to be more than ***999***. In each channel, referring to the compensation group begins from 1 and proceeds up to the preset parameter value, let it be executed either from program at addresses T or D, or from the offset table.
The number of ***common compensation group callable from each channel*** can be given in the parameter N1412 No. of Common Tool Offsets T. Milling machine compensation cannot be referred to from lathe channel.

In each channel, referring to a compensation group from 1 up to the No. of Common Tool Offsets T , indicates the common values, let it be executed either from program at addresses T or D, or from the offset table.

An example:
Let there be 3 channels. Let there be 30 compensations in the channel 1 (No. of Tool Offsets L1=30), 40 compensations in the channel 2 and 60 compensations in the channel 3. Let the quantity of the common compensations be 10 (No. of Common Tool Offsets T=10).

	Channel 1	Channel 2	Channel 3
N001	From N001 up to N010, all the three channels read the common compensations (T1-T10, D1-D10)		
...			
N010			
...	From N011 up to N030, it reads its own compensations (T11-T30, D11-D30)	From N011 up to N040, it reads its own compensations (T11-T40, D11-D40)	From N011 up to N060, it reads its own compensations (T11-T60, D11-D60)
N030			
...			
N040			
...			
N060			

The quantity of compensation groups used in such a way: 10+(30−10)+(40−10)+(60−10)=110.

Distribution of the compensation memory within a channel

Using the offset number, a group of the offset table is selected for the control. The elements of this table are the following:

Group number	**X**		**Y**		**Z**		**R**		**Q**
	Geometry	**Wear**	**Geometry**	**Wear**	**Geometry**	**Wear**	**Geometry**	**Wear**	
1	123.500	-0.234	87.450	-0.129	267.400	-0.036	1	-0.010	3
2									
3									
...									

The tool offset table contains the tool overhang in the directions X, Y and Z, the tool radius (R), and the code of imaginary tool tip (Q).

☞ ***Warning!***

There is no compensation number 00 in the table, offset values on it are always zeros, therefore the command T0 or D0 means cancel of compensation.

Compensations in the directions X, Y and Z and radius compensation (R) have two components: geometry value and wear value.

Geometry value: length/radius of the tool measured. It is a signed number.

Wear value: amount of wears resulting in the course of machining. It is a signed number. In the case of manual data input, the absolute value of the input data is limited to the value given at parameter N1415 Max. Amount of Wear Comp.

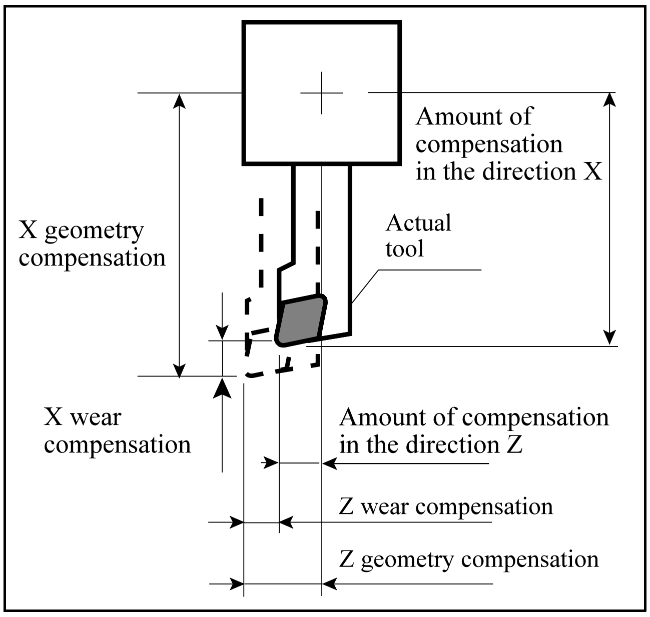


Fig. 15.1-1

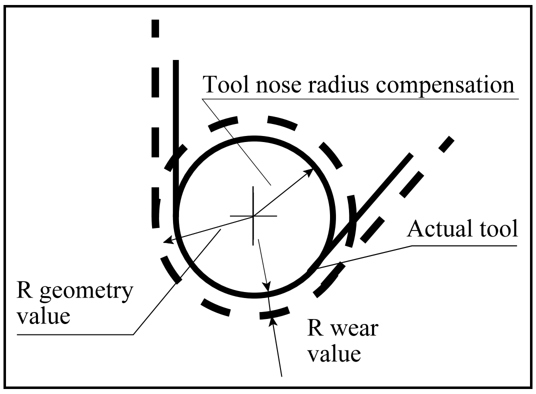


Fig. 15.1-2

Imaginary tool tip: The ***code of the imaginary tool tip (Q)*** is a one-digit number; its value can be 0, 1, ,,, ,9.

The code of imaginary tool tip refers to the direction of position where the imaginary tip of the tool is , viewed from the center point of the tool nose circle. It has to always be given by taking the first and second axes of the selected plane into account. In the case of milling tool, the value of the code Q is always 0 or 9.

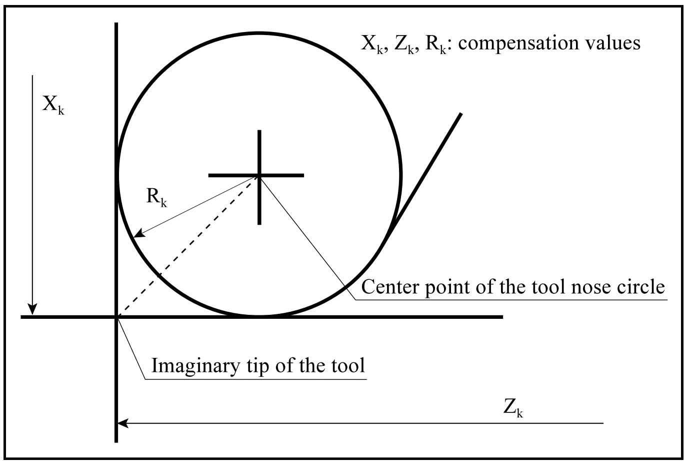

X_k, Z_k, R_k: compensation values
X_k
R_k
Center point of the tool nose circle
Imaginary tip of the tool
Z_k

Fig. 15.1-3

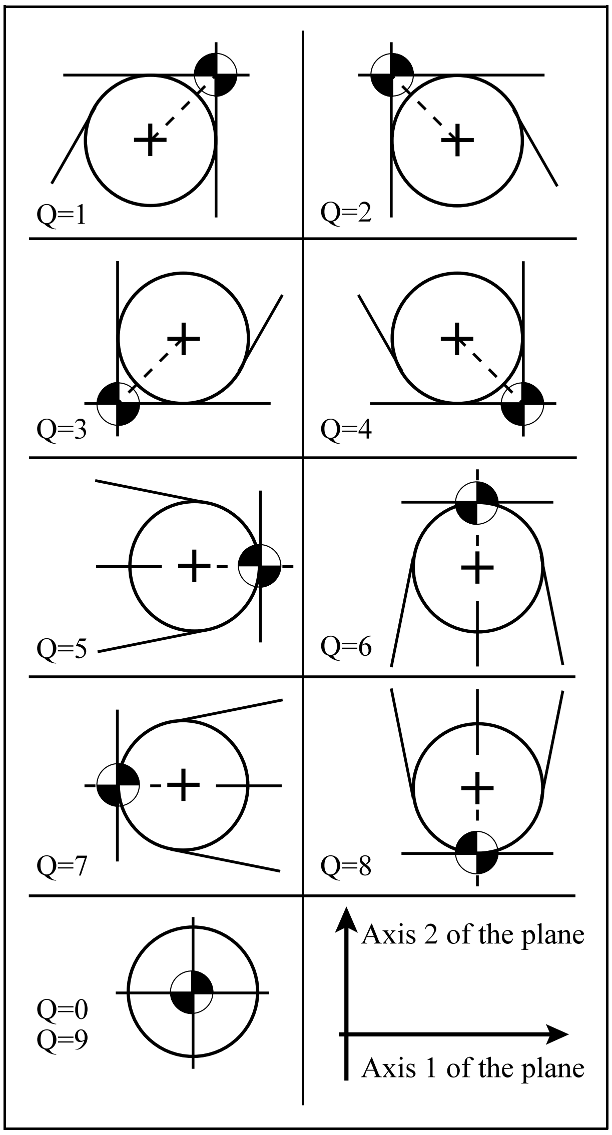


Fig. 15.1-4

The ***compensation code*** called is a ***modal*** one, i.e. the same offset value will be taken into account by the control until it receives another command T or D; it means that if the offset value is read out by a command T or D, modification of the offset table (by programming G10, for example) will not produce effect on the read-out value already.
The offset values in the compensation memory will be retained after power-off.
Tool compensation values can be set or modified by data input from the operator panel, and from the program by the use of the setting command G10.

15.2 The Second Geometry Compensation Memory

In the case of lathes having linear tooling, it is practical to introduce a second geometry compensation memory ***related to length compensation***. The capacity of the second geometry compensation memory is the same as that of the first geometry compensation memory, and in case of referring to geometry length corrections, the values stored in the second geometry compensation memory will be added to the values stored in the first geometry compensation memory by the control, when specific requirements are satisfied.

In the second geometry compensation memory, the position X, Y and Z of the tool holders in the machine coordinate system can be given. Thus, it will be possible to give the real length x, y and z of the tool in the first geometry compensation memory, i.e. to give overhang values measured with external tool measurement device directly in the first geometry compensation memory. Therefore, the offset value taken into account is:

offset = first geometry offset + second geometry offset + wear offset

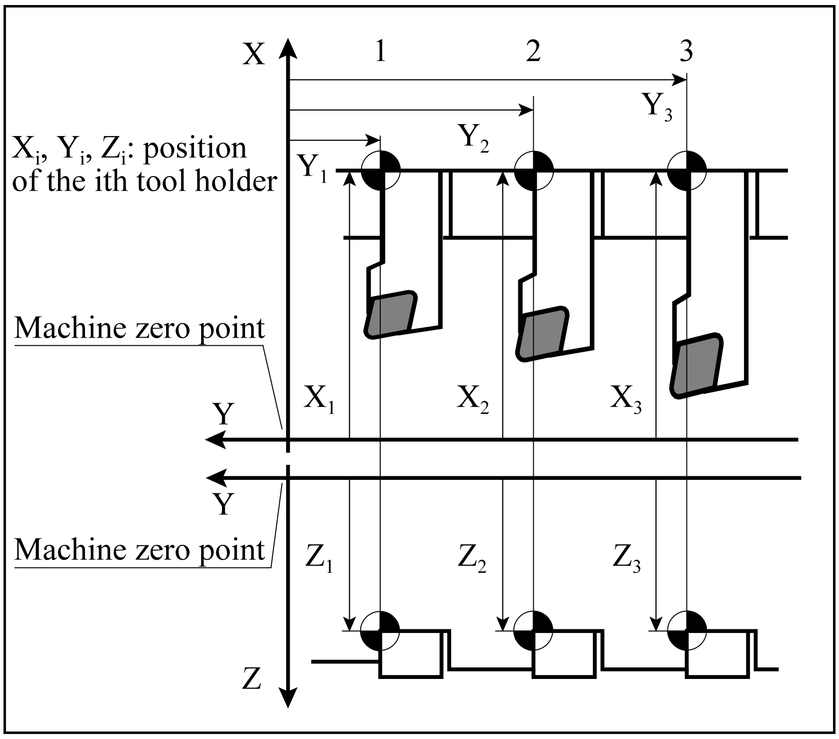


Fig. 15.2-1

Making the second geometry compensation memory to be accessible will be enabled by the bit position #4 SGC=1of the parameter N1414 Comp. Config on Lathes.
Furthermore, it is the PLC program of the given machine that ***enables or disables whether the second geometry offset values will be taken into account or not and with which algebraic sign***. It is the builder of the machine tool who calibrate and manages the second geometry compensation memory.

15.3 Modifying the Tool Compensation Values from the Program (G10)

The command

G10 L P X Y Z R Q

can be used for modifying the tool compensation values from the program. The command G10 is a one-shot command. The meaning of t he addresses and their values is as follows:

L=10: entering geometry value;

L=11: entering wear value.

The compensation group to be modified can be specified at the address P:

P: the number of the compensation group.

The length and radius compensation values can be entered at the addresses X, Y, Z and R:

X: the length compensation value along the axis X;

Y: the length compensation value along the axis Y;

Z: the length compensation value along the axis Z;

R: the radius compensation value.

In the absolute data setting command state **G90,** the value written at the addresses X, Y, Z and R will be put in the appropriate offset register.

In the incremental data setting command state **G91** or in the case of using the operator I, the data written at the address will be added to the content of the appropriate offset register.

Q: the imaginary tool tip (0...9)

The control also accepts the code by specifying L10 and L11.

The data stored in the second geometry compensation memory cannot be modified by the use of the command G10.

15.4 Tool Length Compensation by Code T

At the bit state #0 TCM=0 of the parameter N1414 Comp. Config on Lathes, the code T contains the code of tool compensation too, at the digits of low place value.

It is the parameter N1413 No. of Digits of Offs. No. in T Code which determines how many digits in the code T are needed the compensation.

The reference can be

Tnnnnnnnk

Tnnnnnnkk

Tnnnnnkkk

where:

n...: tool number;

k...: compensation number.

The compensation number calls the geometry compensation together with the wear compensation

It is the case, when at the bit state #1 GTN=0 of the parameter N1414 Comp. Config on Lathes , ***the number of compensation referred at the address T calls geometry compensation together with the wear compensation.***

In the case of the reference Tnnkk:

– the control transmits the value nn to the PLC;

– the control adds the value of the ***kk geometry register*** together with the value of the ***kk wear register***, and the sum will be the called value of the compensation:

Compensation= Geometry_{kk} + Wear_{kk}

The command

Tnn00

cancels the compensation.

The tool number calls the geometry compensation, the compensation number calls the wear compensation

It is the case, when at the bit state #1 GTN=1 of the parameter N1414 Comp. Config on Lathes , ***the number of the tool referred at the address T calls geometry compensation, and the number of the compensation referred calls the wear compensation.***

In the case of the reference Tnnkk:

– the control transmits the value nn to the PLC;

– the control adds the value of the ***nn geometry register*** together with the value of the ***kk wear register***, and the sum will be the called value of the compensation:

Compensation= Geometry_{nn} + Wear_{kk}

The solution above raises two problems. The first one: from which register (nn or kk) the imaginary tool tip, i.e. the ***value of Q*** has to be taken by the control; the second one: how the reference ***Tnn00*** has to be interpreted by the control.

If the state of the bit #2 QGW of the parameter N1414 Comp. Config on Lathes is:

=0: ***the imaginary tool tip (Q) will be taken from the wear code register*** by the control;

=1: ***the imaginary tool tip (Q) will be taken from the geometry code register*** by the control

If the state of the bit #3 ZCG of the parameter N1414 Comp. Config on Lathes is:

=0: the ***compensation number 0*** (Tnn00) will not delete the geometry compensation, but it deletes the ***wear compensation*** only;

=1: the ***compensation number 0*** (Tnn00) will delete both the geometry compensation and the ***wear compensation***.

Due to the code T the coordinate system will be shifted by the compensation values X_k, Y_k, Z_k belonging to the compensation group specified in the code T. This means, that the compensation values in the direction X_k, Y_k, Z_k belonging to the selected compensation group will be subtracted from the actual positions X, Y, Z. From that moment, not the coordinates of the reference point of the tool holder, but the coordinates of the imaginary nose of the tool will be displayed on the screen. Hereafter, compensation value means the sum of the geometry compensation and wear compensation.

For example, due to the following command lines

```
T0
N10 G0 G90 X700 Z350
N20 T202
N30 X300 Z150
```

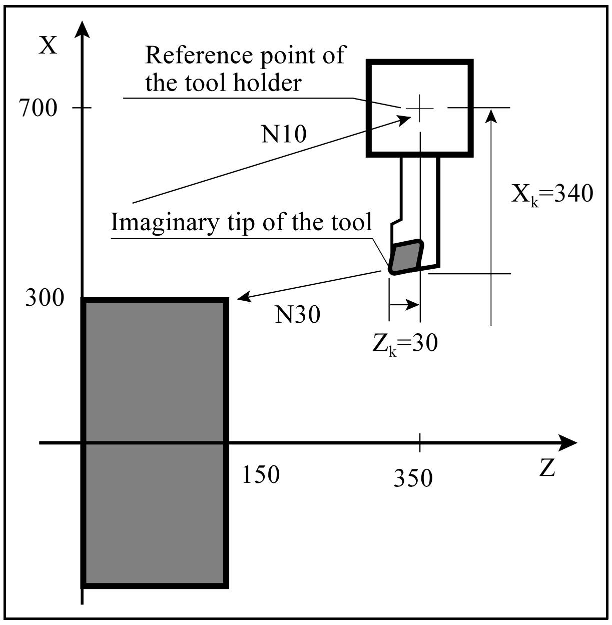


Fig. 15.4-1

the control moves the reference point of the tool holder to the point of coordinates

X700; Z350

according to the block N10. In the block N20, it executes the tool change, and it subtracts the compensation values X_k=340, Z_k=30 belonging to the compensation group 2 from the actual position. Now, the position display will show the following values:

X=700–340=360; Z=350–30=320

No motions will happen in the block N20. In the block N30, the control will already move the imaginary nose of the tool to the programmed point X300; Z150, i.e. the displacement will be

X=300–360= –60; Z=150–320= –170.

If the code T is given ***together with the motion block***, motion to the block-end position will already be carried out in accordance with the compensation value specified by the code T. The tool change, however, will be executed parallelly with or at the end of the motion if the code T contains tool change command, too. (The time of tool change is defined by the builder of the machine tool .)

Let us return to the example above:

```
T0
N10 G0 G90 X700 Z350
N20 X300 Z150 T202
```

Now, the block N30 is left and the commands of the blocks N20 and N30 are merged. The control will already move the imaginary nose of the tool to the point of coordinates

X=300, Z=150

as in the case of the block N30 in the previous example.

If ***compensation number only is programmed at code T***:

```
N10 G0 G90 X700 Z350
N20 X300 Z150 T2
```

tool change will not be executed, and compensation group 2 only was called in the block N20.

Cancelling the length compensation

The specific role of the compensation group 0 is cancelling compensation.

Due to the command

Tnn00 or

T0

the length compensation will be cancelled (nn is an arbitrary tool number). The process is exactly inverse of the calling the compensation. This means, that the compensation values X_k, Y_k, Z_k which were valid previously will be added to coordinates X, Y, Z of the imaginary nose of the tool, and from that moment, the coordinates of the reference point of the tool holder will be displayed on the screen. If cancelling the compensation is done together with motion block, the reference point of the tool holder will be sent to the programmed end point of the block by the control.

For instance, in the example

```
N10 X180 Z120 T202
N20 X200 Z180
N30 X280 Z210 T200
```

in the block N10 compensates the motion with the offset called in the comparison with

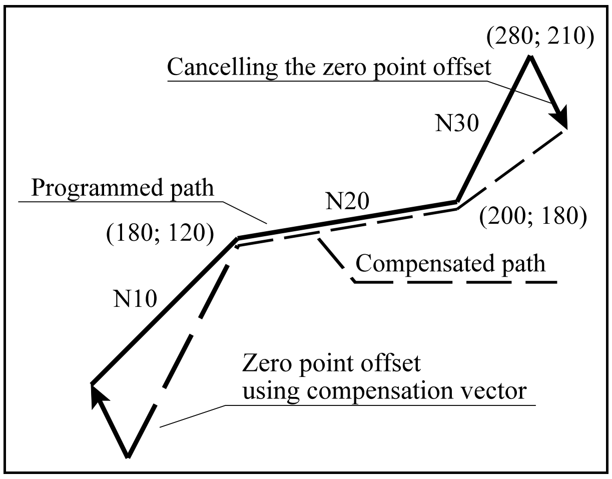


Fig. 15.4-2

the programmed one; and in the block N30 it cancels the compensation, namely the zero point offset set in the block N10.

15.5 Tool Length Compensation by Code G (G43.7, G49)

At the bit state #0 TCM=1 of the parameter N1414 Comp. Config on Lathes, calling the length compensation will be executed by the code G.
The command

G43.7

starts the tool length compensation. If it is programmed together with axis address, positioning to the given point will already be carried out by the control using the new compensation.
Meaning of the address D: the value of the tool length compensation is taken from the compensation group specified at this address.

In the case of ***absolute data specification*** the command

```
      G43.7 G0 G90 X100 Z10 D5
```

moves the tip of the tool to the point of coordinates X100 Z10, taking the compensation D5 into account.

In the case of ***incremental data specification*** the command

```
      G43.7 G0 G91 X20 Z10 D1
```

shifts the starting point by the compensation value, and then the incremental displacement will be counted from here, from the starting point compensated.
Accordingly, due to the command

```
      G43.7 G91 X0 Y0 Z0 D2
```

the axes X, Y and Z do not execute motion; only the position will get the value adequate to the new compensation.

The effect of ***G43.7*** is ***modal*** until the control receives another command from this group. If the compensation value is changed by calling a ***new address D***, the old value will be cancelled, and the new value will be validated:

```
      G43.7 X120 Z100 D1  (it moves to the machine position X150
                          Z=110)
      ...
      X120 Z100 D2 (it moves to the machine position X170 Z=120)
```

The state of G43.7 is hereditary by the block X120 Z100 D2, therefore the new compensation D2 will be taken into account.
D0 cancels the compensation. Continuing the example above:

```
      G43.7 X120 Z100 D2  (it moves to the machine position X170
                          Z=120)
      ...
      X120 Z100 D0 (it moves to the machine position X120 Z100)
```

The command

G49

will switch the tool length compensation off on all the axes either by motion if axis address is also programmed in the block, or by transformation, if motion is not programmed in the block.

At power-on, at reset or after the program end, the bits #5 G43 és #6 G44 of the parameter N1300 DefaultG1 determine code G43.7 is valid. If both bits are 0, then code G49 is valid.

15.6 Tool Nose Radius Compensation (G40, G41, G42)

It will not be possible to turn exact conical surface or circle arc if tool length compensation only is applied. In the case of using tool length compensation, the imaginary nose of the tool is guided by the control along the programmed path. Since the nose of every tool has a smaller or bigger fillet, accurate machined surface can only be achieved when motions are parallel with the axes, as it is shown in the figure. In the case of conical or curved surfaces, the diameter of the workpiece will be greater than the programmed value, in every point. The difference is illustrated with dashed line in the figure.

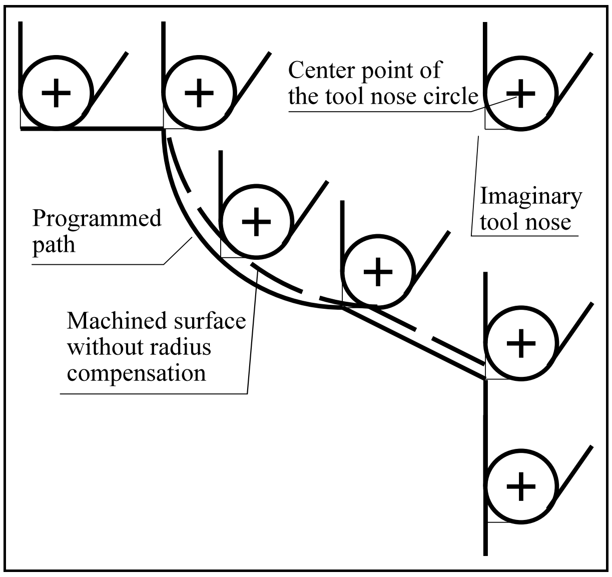


Fig. 15.6-1

In order that an optional contour can be turned accurately, and the contour points corresponding to the drawing can be given in the program independently of the radius of the tool nose circle, ***the center point of the tool nose circle has to be guided by the control parallelly with the programmed contour in a distance of the radius from it***. The distance of the center point of the tool nose circle from the programmed contour during guidance is determined by the control according to value of the ***tool nose radius compensation*** entered on the compensation number ***called at the address T or D***.

The position of the imaginary tool nose is registered by the control before moving to the contour. ***The code of imaginary tool tip*** given in the tool offset table at the address Q is necessary to decide, in which direction is the center point of the tool nose circle in comparison with the imaginary nose. In accordance with the code of imaginary tool tip, the value of the tool nose radius (R) will be added/subtracted by the control to/from the position of the imaginary nose in the directions X and Z (position A in the figure). Then, the control guides ***the center point of the tool nose circle*** to the endpoint of the R-long ***compensation vector*** placed perpendicularly on the starting point of the programmed path (position B in the figure). After that, the control guides the center point of the tool nose circle parallelly with the programmed contour in the distance R from it.

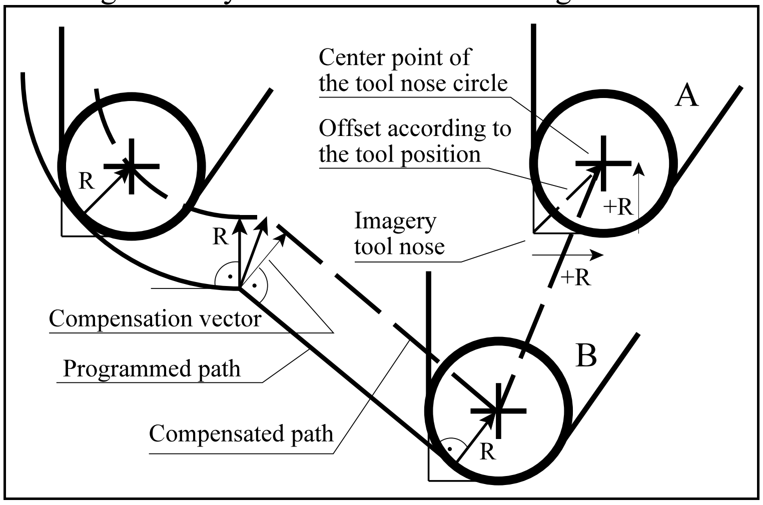


Fig. 15.6-2

The ***compensation vector*** is a ***two-dimensional vector*** recalculated by the control in each block, and the programmed displacements are modified by the control with the compensation vectors effective at the beginning and at the end of the block. The length and the direction of the compensation vectors obtained depends on the compensation value called at the address T and the geometry of the transition between the two blocks.

The control computes the compensation vectors ***in the plane selected by the commands G17, G18 and G19***.This is the plane of tool nose radius compensation. Motions out of this plane are not affected by the radius compensation. For example: if, in the state G18, the plane X, Z is selected, the compensation vectors will be computed in the plane X, Z.
It is not permitted to change compensation plane while tool nose radius compensation is being computed. Should it is tried, the control will send an error message.
In the case of defining the compensation plane not along the axes being in the main plane, secondary axes have to be defined in the parameter field as parallel axes. For example, if U is defined as parallel axis and tool nose radius compensation should be used in the plane Z, U, the plane will have to be assigned by specification G18 U__ Z__ .

Computing the tool nose radius compensation can be switched on and off from the program:

G40: **cancelling** calculation of the tool nose radius compensation

G41: calculation of the tool nose radius compensation **to the left according to the motion direction**

G42: calculation of the tool nose radius compensation **to the right according to the motion direction**

The commands ***G40, G41, G42*** are ***modal*** ones. After power-on, at the program end and due to reset the control comes to the state G40.

The command G41 or G42 starts the compensation calculation. The programmed contour is tracked by the tool from the left in the state G41, and from the right in the state G42, according to the motion direction. ***The applied values of tool nose radius compensation must be given at the address T or D. Compensation calculation is carried out for the interpolation motions G00, G01, G02, G03***.

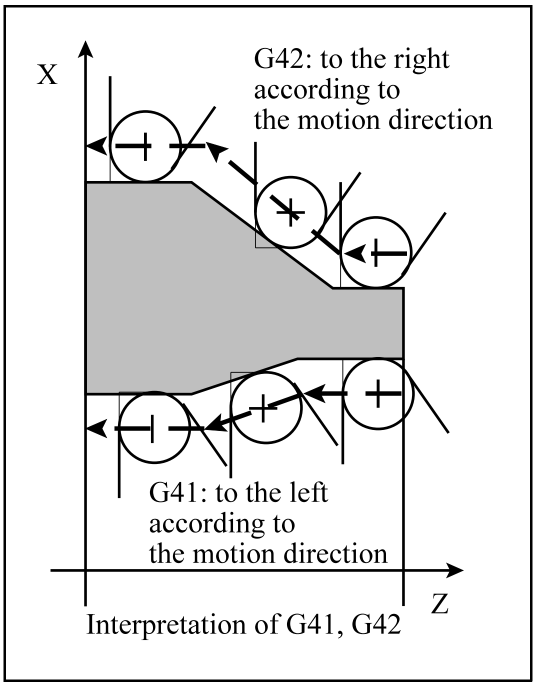

X
G42: to the right according to the motion direction
G41: to the left according to the motion direction
Z
Interpretation of G41, G42

Fig. 15.6-3

The findings mentioned above are valid in the case of positive tool nose radius compensation. However, ***the value of tool nose radius compensation can be negative, too***. It has practical meaning, for example, in the case when the same subprogram is to be used for contouring a female workpiece and then a male workpiece fitting together. Another way of doing this is that G41 is used for machining the female workpiece and G42 is used for machining the male workpiece. But this changeover will not have to be edited into the program if the female one is machined with positive radius compensation and the male one is machined with negative radius compensation, for example. In this case, the path of the tool center point will be reversed by the programmed G41 or G42:

	Radius compensation: **positive**	Radius compensation: **negative**
G41	from the left	from the right
G42	from the right	from the left

☞ ***Note***: For the sake of simplicity, hereafter, positive tool nose radius compensation is always assumed in descriptions and figures.

The commands ***G40*** or ***Tnn00***, ***T0***, ***D0*** cancel the compensation calculation. The difference between the two commands is that the command T0 uses only compensation vector of 0 length, and leaves the state G41 or G42 unchanged. If, after that, a new address Tnnmm (mm≠0), D is referred to, the compensation vector will be calculated using the new tool nose radius, depending on the state G41 or G42.
But, if the command G40 is used, the control will stop calculation of compensation vectors.
There are specific rules for starting or cancelling the tool nose radius compensation; they will be detailed in the following.

Radius compensation commands are executed by the control in automatic or manual data input mode only. In manual mode, it is not effective in case of single blocks. The reason for this is as follows. In order that the control can calculate the compensation vector in the endpoint of a block, it is necessary for the control to read in the next block containing the motion in the selected plane. The compensation vector depends on the transition between the two blocks. It is apparent that preliminary processing of several blocks is required for calculation of the compensation vector.

Before dealing with details of the compensation calculation, an auxiliary data is to be introduced. It is the angle α between the tangent lines drawn to the meeting point of two curve segments, i.e. to blocks. The direction of the angle α depends on whether the contour is compassed from the left or from the right.
The control selects the strategy of deflecting at the intersection points according to the angle α. If α>180°, i.e. the tool works inside, the control will compute an intersection point between the two segments. If α<180°, i.e. the tool moves outside, the control will insert additional straight segments for moving.

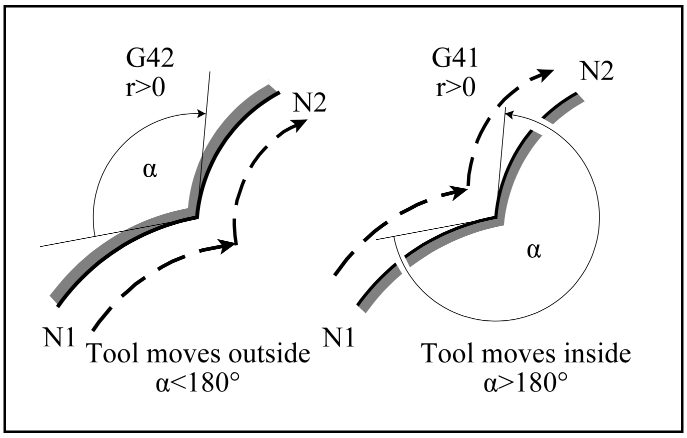


Fig. 15.6-4

Hereafter, this manual deals with cases of calculation of tool nose radius compensation mainly from the point of view of turning operations. For milling operations, cases of calculation of radius compensation are dealt with in the book NCT2xxM Control for Milling Machines and Machining Centers Programmer's Manual.

15.6.1 Start up of the Tool Nose Radius Compensation. Moving to the Contour

Due to the command G41 or G42, the control enters in the mode of tool nose radius compensation from the state G40. It takes the value of compensation from the compensation cell specified at the address T.
The control comes to the state ***G41*** or ***G42*** in the block of ***positioning G0*** or in the block of ***linear positioning G1*** only.
If start up of compensation calculation is attempted in circular interpolation (G2, G3), the control will send an error message.
The ***strategy of moving to the contour*** is chosen by the control in the only case of change over ***from the state G40 to the state G41 or G42***.
The basic cases of starting up the compensation according to the angle α and the possible transitions (linear - linear and linear - circular) are illustrated below. The figures are drawn for the case of G42 and ***positive radius compensation*** is assumed.
☞ *Note*: Here and hereafter, meanings of the marks in the figures are the following:

r: value of radius compensation;
L: linear segment;
C: circle arc;
S: in the block by block mode, the point of stop;
dash line: the path of the center point of the tool nose;
continuous line: the programmed path.

The basic cases of starting up the tool nose radius compensation:

```
G40                     G40
G42 G1 X_ Z_            G42 G1 X_ Z_
X_ Z_                   G2 X_ Z_ R_
```

Positioning to an inner corner: 180°<α<360°

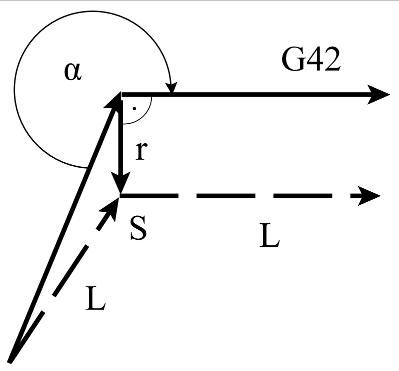
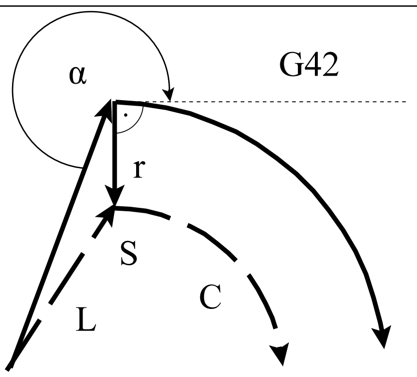
Linear to linear	Linear to circular
	

Fig. 15.6.1-1

Positioning to an outer corner at an obtuse angle: 90°≤α≤180°

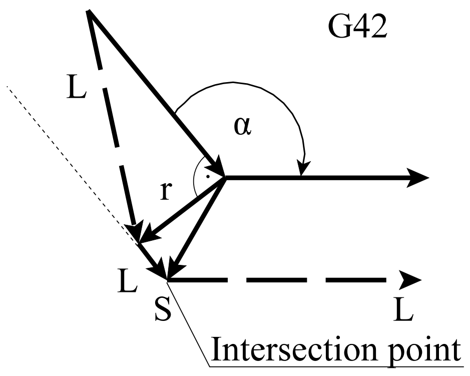
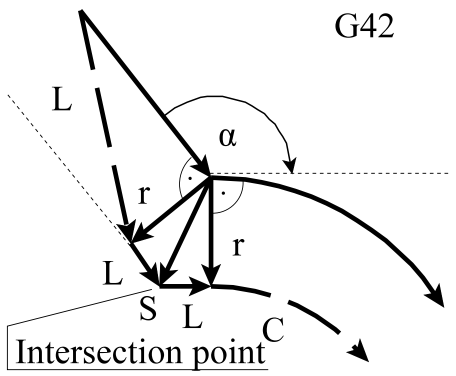
Linear to linear	Linear to circular
	 G42 L α r r L S L C Intersection point

Fig. 15.6.1-2

Positioning to an outer corner at an acute angle: 0°≤α<90°

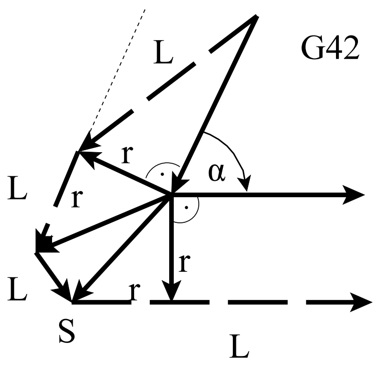
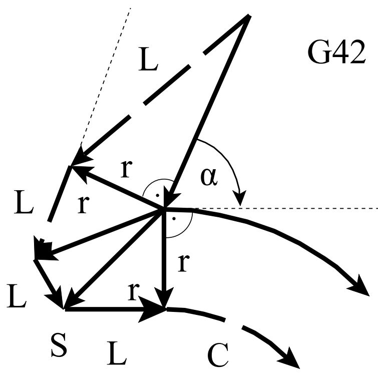
Linear to linear	Linear to circular
	

Fig. 15.6.1-3

An example:
The program below is an example for correct starting up of tool nose radius compensation at the code of imaginary tool tip Q3.
The tool nose radius compensation is started up by the block N110. The control positions to the beginning of the block N110 with the imaginary nose of the tool, and then, at the end of the block, it contacts the circle of the tool nose with the starting point of the block N120:

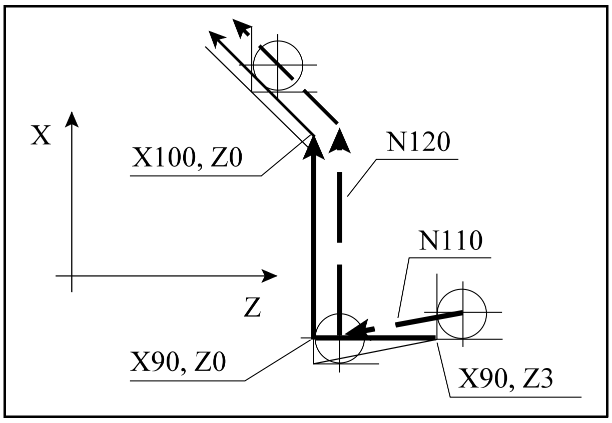


Fig. 15.6.1-4

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
```
`N110 G1 G42 Z0`
```
N120 X100
N130 X110 Z-5
N140 Z-25
N150 G2 X130 Z-35 R10
N160 G1 X140
N170 Z-45
N180 G40 X160
N190 G0 X200 Z50
...
```

Starting up the tool nose radius compensation without programming of motion:
If ***tool nose radius compensation is started up*** (G41, G42) in a block in which there are ***none of the addresses of the axes belonging to the selected plane*** (as in the example below, neither the address X nor the address Z is given in the block N110), motion in the block N110 will not be executed. Motion to the endpoint of the block N120 will be executed in such a way that the control calculates intersection point between the blocks N120 and N130.
It is not good, because the point X90 Z0 will not be machined! Correction can be done by merging the blocks N100 and N120.

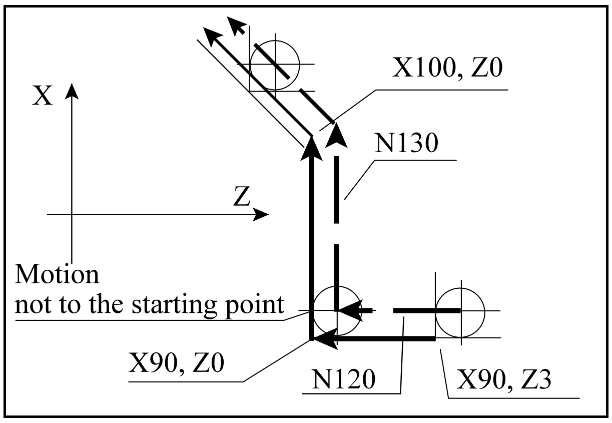


Fig. 15.6.1-5

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
```
`N110 G1 G42`

```
N120 Z0
N130 X100
N140 X110 Z-5
N150 Z-25
N160 G2 X130 Z-35 R10
N170 G1 X140
N180 Z-45
N190 G40
N200 X160
N210 G0 X200 Z50
...
```

If ***tool nose radius compensation is started up*** (G41, G42) in a block in which ***there is no displacement along none of the axes belonging to the selected plane*** (as in the example below, in the block N110 the address Z is filled, but because of the incremental 0 belonging to it there is no motion), there will be displacement with a value of tool nose radius in the block N110. In this case, the control guides the center point of the tool nose while calculating intersection point between the blocks N120 and N130.It is not good, because the corner X90 Z0 will not be machined! Correction can be done by merging the blocks N110 and N120.

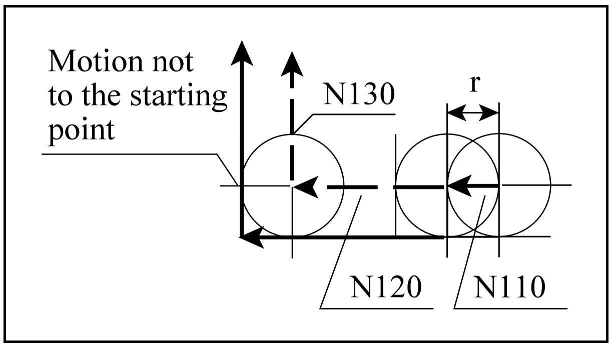


Fig. 15.6.1-6

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
N110 G1 G42 G91 Z0
N120 G90 Z0
N130 X100
N140 X110 Z-5
N150 Z-25
N160 G2 X130 Z-35 R10
N170 G1 X140
N180 Z-45
N190 G91 G40 X0
N200 G90 X160
N210 G0 X200 Z50
...
```

There is no displacement in the block succeeding the start up of the tool nose radius compensation

If, in the block succeeding the start up of the tool nose radius compensation, there is displacement 0 in the selected plane, as in the block N120 of the program below, the control will guide the circle of the tool nose to the endpoint of the block carrying out the start up (N110). Then it guides

the circle of the tool nose to the starting point of the block carrying out the succeeding motion (N130), it moves on up to this point, and then moves the tool onward, in parallel.
This can cause overcutting on the workpiece, so these cases have to be avoided! Correction can be done by cancelling the block N120.

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
N110 G1 G42 Z0
N120 Z0
N130 X100
N140 X110 Z-5
N150 Z-25
N160 G2 X130 Z-35 R10
N170 G1 X140
N180 Z-45
N190 G91 G40 X0
N200 G90 X160
N210 G0 X200 Z50
...
```

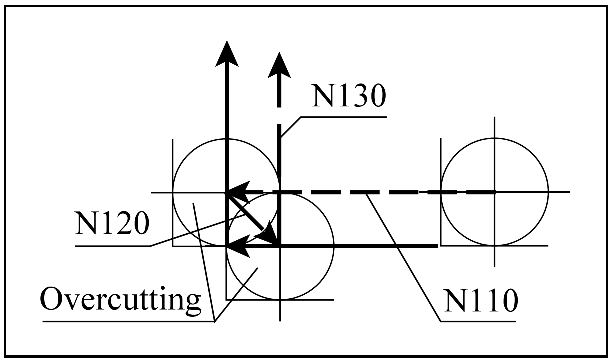


Fig. 15.6.1-7

Starting up the tool nose radius compensation with calculation of intersection point

If, in the block carrying out start up (G41 or G42), a value is assigned to I, J and K, but only to those that are in the selected plane (for example, to I and K in the case of G18), the control will move to the intersection point defined by the succeeding block and by I, J and K, taking the tool nose radius compensation into account. The value of I, J and K is always incremental, and the vector defined by them points at the endpoint of the that block in which it was programmed. It is a useful thing in the case of positioning to an inner corner, for example.

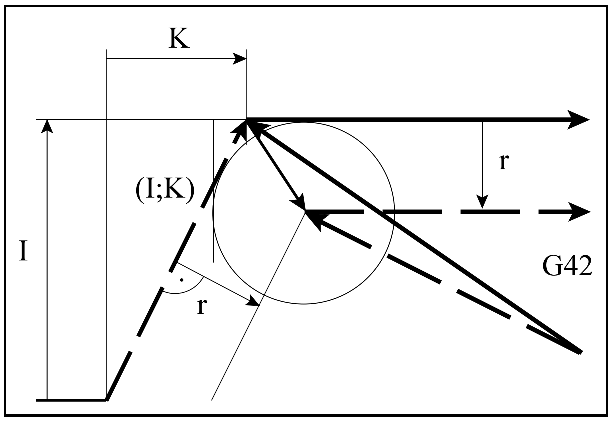


Fig. 15.6.1-8

An example:

A hexagon with side length of 100 mm is to be milled internally. The tool has to be positioned to the corner point with the coordinates of X100 Y0.The block N50 of the program below positions to the contour using calculation of intersection point, with specification of I and J. The coordinates of I and J were calculated on the basis of the displacement executed in the block N110 of the program.

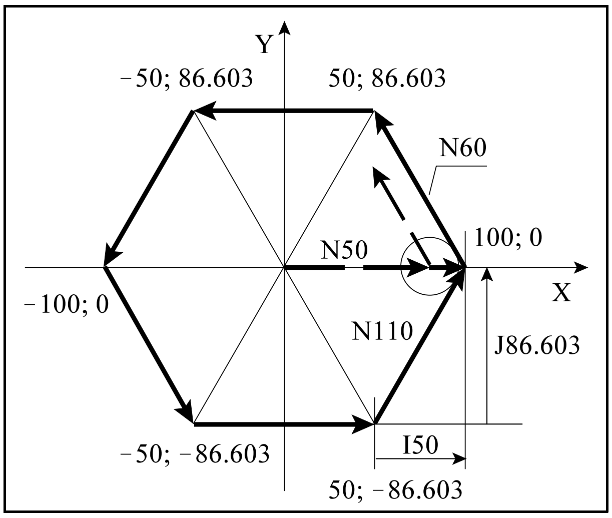


Fig. 15.6.1-9

```
N10 G54 G17 M3 S300
N20 G0 X0 Y0 Z100
N30 Z5
N40 G1 Z-5 F1000
N50 G41 G0 X100 Y0 I50 J86.603 D1
N60 G1 X50 Y86.603
N70 X-50
N80 X-100 Y0
N90 X-50 Y-86.603
N100 X50
N110 X100 Y0
N120 G40 G0 X0 I-50 J86.603
N130 M30
```

Cases of positioning using intersection point

In the case of specification I, J and K, the control always calculates intersection point regardless of whether inner or outer corner is to be machined.

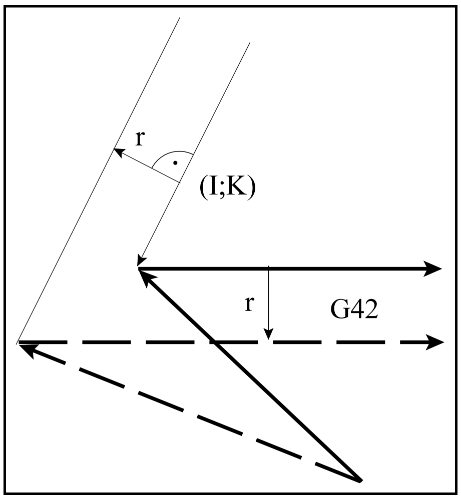


Fig. 15.6.1-10

If the control does not find intersection point, it will position to the starting point of the succeeding block at right angles.

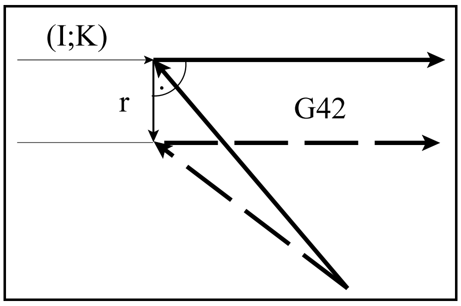


Fig. 15.6.1-11

15.6.2 Calculation of Tool Nose Radius Compensation in Offset Mode

In the course of calculation of tool nose radius compensation in offset mode, the compensation vectors will be calculated continuously between the blocks G0, G1, G2 and G3. In order that these vectors can be calculated, the blocks have to be continuously read ahead.
The control reads ahead + 2 blocks, the number of which is given in the parameter N1404 BK No. Interf. This means, that the compensation calculation will remain continuous yet if between two motion blocks belonging to the selected plane there is an other block numbered in parameter, for example function, dwell, motion outside of the plane etc.
Continuance may be interrupted by those codes G and functions which enforce buffer emptying, i.e. suspend reading ahead of the blocks.

The basic cases of calculation of tool nose radius compensation in offset mode:

Calculation of intersection point in the case of inner corners: 180°<α<360°

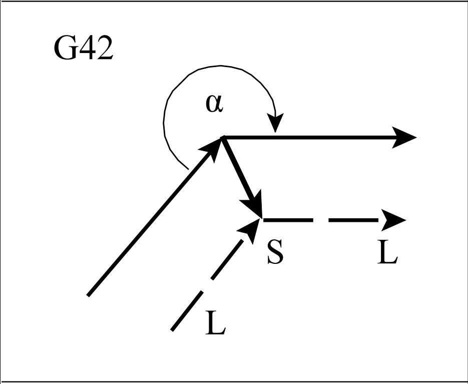
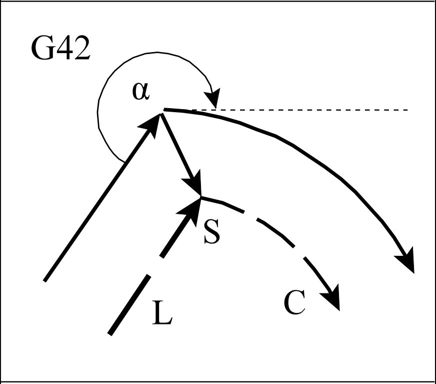
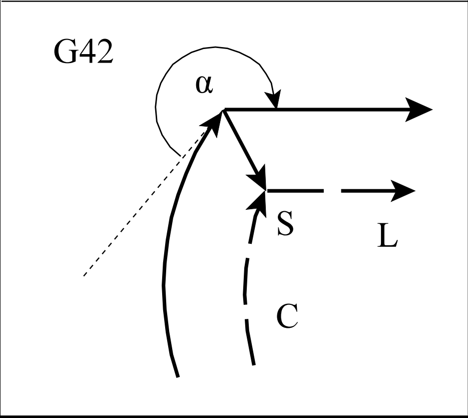
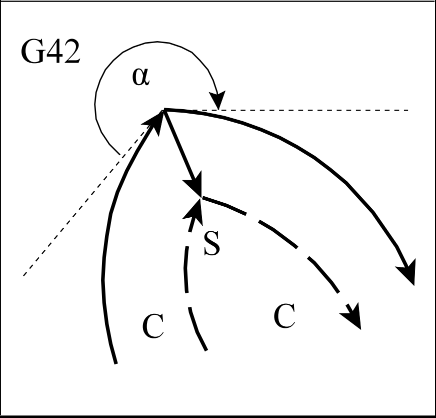
Linear to linear	Linear to circular
	
Circular to linear	**Circular to circular**
	

Fig. 15.6.2-1

Going around outer corners having obtuse angle: 90°≤α≤180°

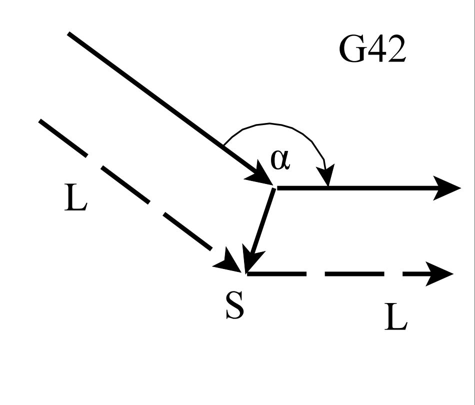
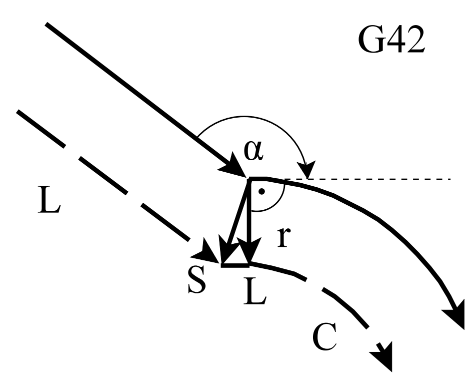
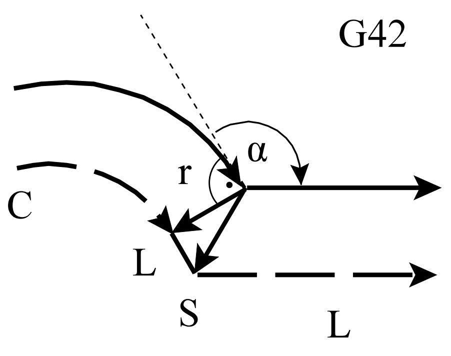
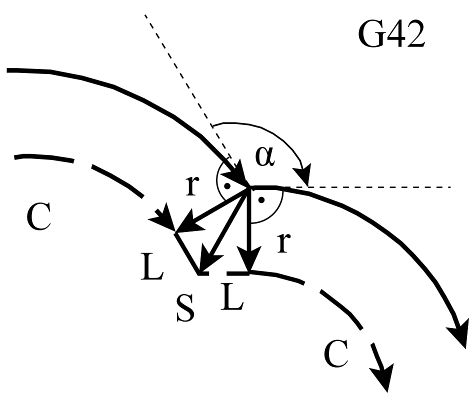
Linear to linear	Linear to circular
	
Circulat to linear	Circular to circular
	

Fig. 15.6.2-2

Going around outer corners having acute angle: 0°≤α<90°

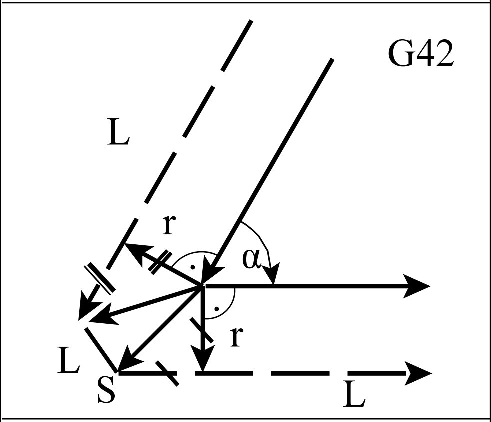
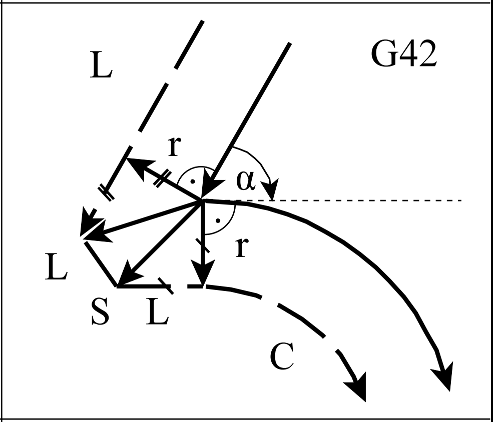
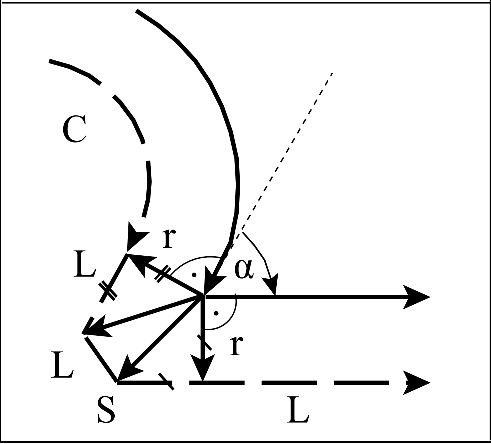
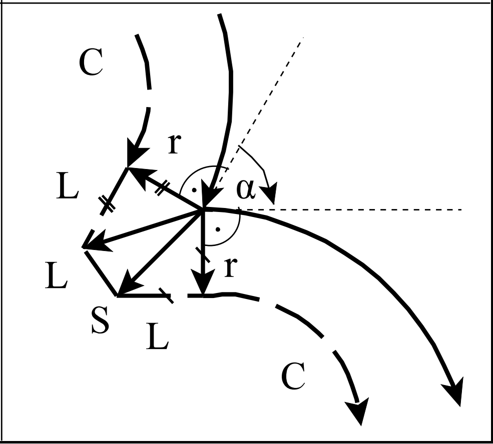
Linear to linear	Linear to circular
	
Circular to linear	**Circular to circular**
	

Fig. 15.6.2-3

There is no intersection point in case of inner corners

It can happen, that there will not be intersection point at certain values of tool radius. In this case, the control will stop during execution of the previous block and will send error message '2047 No intersection G41, G42'.

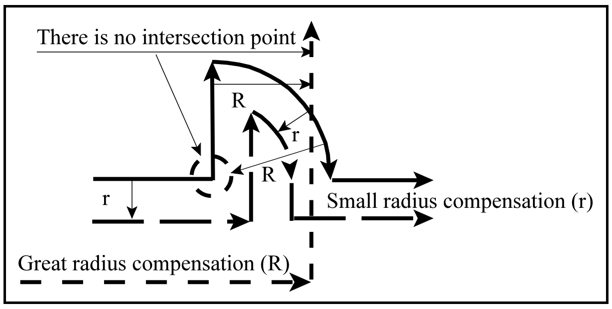

There is no intersection point
R
r
R
r
Small radius compensation (r)
Great radius compensation (R)

Fig. 15.6.2-4

Zero displacement in the selected plane takes place

If, when G41 or G42 is effective, zero displacement in the selected plane is programmed in one of the blocks or zero displacement takes place as in the the block N120 of the example below, the following will happen. The control lets fall perpendicular vectors to the endpoint of the previous block (N110) and to the starting point of the succeeding block (N130) the length of which is equal to the tool nose radius compensation, and then it connects these two vectors by linear interpolation. In these cases, increased attention is required because unintended overcutting or, in case of circle, distortion can be caused.

For example:

```
    ... G91 G18 G42 ...
    N110 G1 X100 Z40
    N120 Z0
    N130 Z90
    N140 X-40 Z50
    ...
```

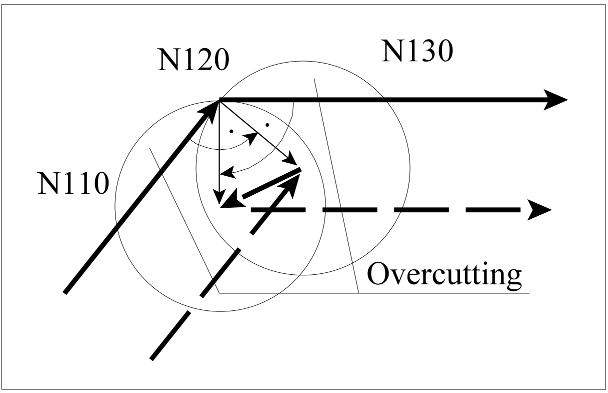


Fig. 15.6.2-5

Tool nose radius compensation for a spiral or a circle with variable radius

If tool nose radius compensation is used for a spiral or a circle with variable radius, the control calculates compensation vector(s) in the starting point of the circle for an imaginary circle, the radius of which is equal to the starting point radius of the programmed circle (in the figure R1=50) and the center point of which coincides with the center point programmed (X0, Z0). The control calculates compensation vector(s) in the endpoint of the circle for an imaginary circle, the radius of which is equal to the endpoint radius of the

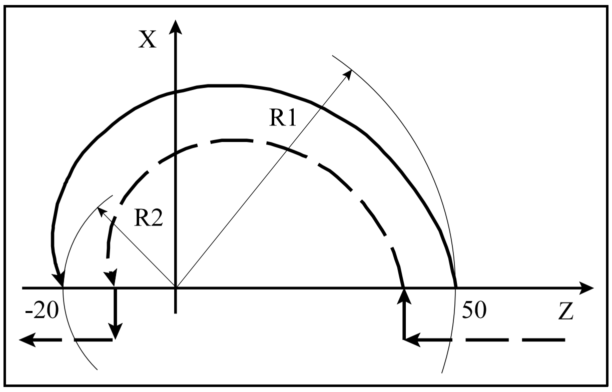


Fig. 15.6.2-6

programmed circle (R2=20) and the center point of which coincides with the center point of the circle programmed.
An example:

```
G0 G18 G41
G1 Z50
G3 Z-20 K-50 L1
G1 Z-30
...
```

Omitting the corner movements

When moving around acute-angled or obtuse-angled corners, two or more compensation vectors can be produced. If their endpoints almost coincide with each other, there barely will be motion between the two point.
In the case, when the distance between the two vectors on both axes is less than the value set in the parameter N1405 DELTV, the vector shown in the figure will be omitted and the path of the tool will be changed as it is illustrated in the figure.

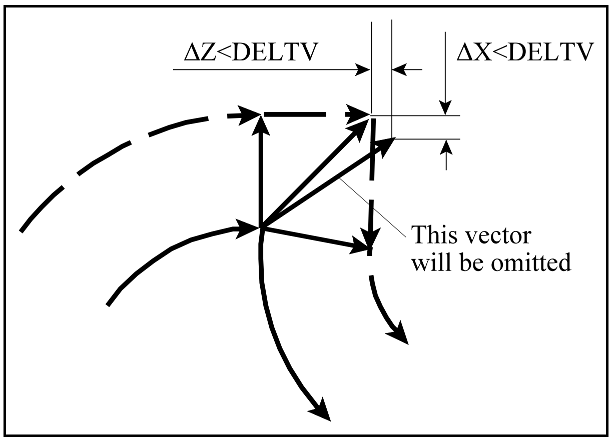


Fig. 15.6.2-7

☞ *Note*: If the value of the parameter DELTV is unduly great while moving around outer acute-angled corners, the corner can be damaged by the tool!

In the selected plane, there is no motion command in several consecutive blocks

In order that the control will manages tool nose radius compensation in a proper way, for example, it can be able to calculate intersection point between the endpoint of the block read in and the starting point of the succeeding contour block, ***the blocks have to be read ahead and preprocessed***. The preprocessed blocks get into the block buffer.
In practice, it could be necessary to program a block not containing motion or a block containing motion performed not in the selected plane, in between two block of planar motion. These, for example, can be the following:

functions: M, S, T
dwell: G4 P
motion performed outside of the selected plane: (G18) G1 Y
subprogram call: M98 P, etc.

This means, that if there are any other blocks between two motion blocks belonging to the selected plane, for example, functions, dwell motion outside of the plane etc., the compensation calculation will remain continuous yet until the block buffer will become full.
When the ***buffer becomes full***, the control will send the error message
‘2090 Unable to continue radius compensation. Buffer full’
at the beginning of the last motion block belonging to the selected plane.

Buffer emptying function is programmed when G41 or G42 is effective

Continuance of compensation calculation, i.e. ***reading ahead*** of the blocks ***will be interrupted*** by those codes G and functions (for example certain functions M etc.) which enforce ***buffer emptying***.

When in the course of reading ahead the control reads in such a code, it suspends reading ahead of further blocks and waits until the block buffer becomes empty, i.e. until all the blocks in the buffer are executed. It results in ***suspending calculation of tool nose radius compensation*** by the control. Then, the control executes the function, and after that it begins reading in and buffering the succeeding block.

The codes G suspending calculation of tool nose radius compensation are the following:

G22, G23,
G54-G59, G54.1, ...,
G52, G92,
G53,
G28, G30

The function T, being related either to tool change or to calling the compensation, also suspends compensation calculation:

Tnnkk, Tnn00, Tkk

The functions M suspending calculation of tool nose radius compensation are the following:

M0, M1, M2, M30

In addition to the functions M above, functions M and groups of codes M executing buffer emptying can be assigned in parameters, too. Functions S and auxiliary functions (A, B, C, U, V, W) besides functions M can also be assigned for buffer emptying.

The functions executing buffer emptying are determined by the the builder of the machine tool, and they are contained in the manual of the machine!

In the case, when G41 or G42 is effective, and the abovementioned codes G or functions are programmed between two motion block, the control cancels the compensation vector at the endpoint of the previous block, executes the command and then restores the vector at the endpoint of the succeeding motion block.

For example:

```
...G91 G18 G41...
N110 G1 X-100 Z80
N120 G92 X0 Z0
N130 X100 Z80
...
```

In the example above, the control empties the buffer. The situation is similar to the cases of the other buffer emptying codes, too.

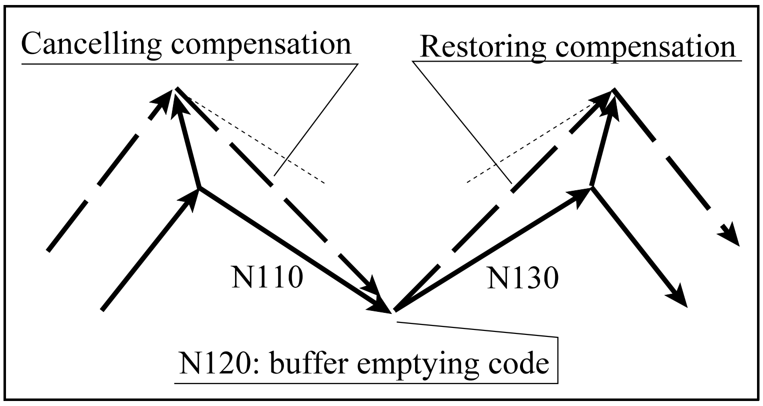


Fig. 15.6.2-8

Conditional block skip is programmed when G41 or G42 is effective

At the bit state ***#4 CBB=0*** of the parameter N1337 Execution Config, the conditional block command (blocks beginning with slash /) ***suppresses*** reading the block ahead. In this case, when the ***G41, G42*** is effective, the contour ***will be distorted***, however, for effectiveness, it is enough ***to turn*** the conditional block switch ***during execution of the previous block***.

At the bit state ***#4 CBB=1*** of the parameter N1337 Execution Config, the conditional block command (blocks beginning with slash /) ***does not suppress*** reading the block ahead. In this case, when the ***G41, G42*** is effective, the contour ***will not be distorted***, however, for sure effectiveness, the conditional block switch ***has to be set before execution of the program***.
For the effect of the parameter setting, please see the subchapter Conditional block skip.
An example:

```
...G91 G18 G41...
N110 G1 X-100 Z80
/ N120 S2500
N130 X100 Z80
...
```

In the example above, the block N120 is conditional.
If CBB=0, the control will cancel compensation at the end of the block N110 and will recover it in the block N130.
If CBB=1, the control will not suspend the reading ahead, the calculation of compensation will be continuous.

In the state of G41, G42 it is recommended to avoid programming a conditional block.

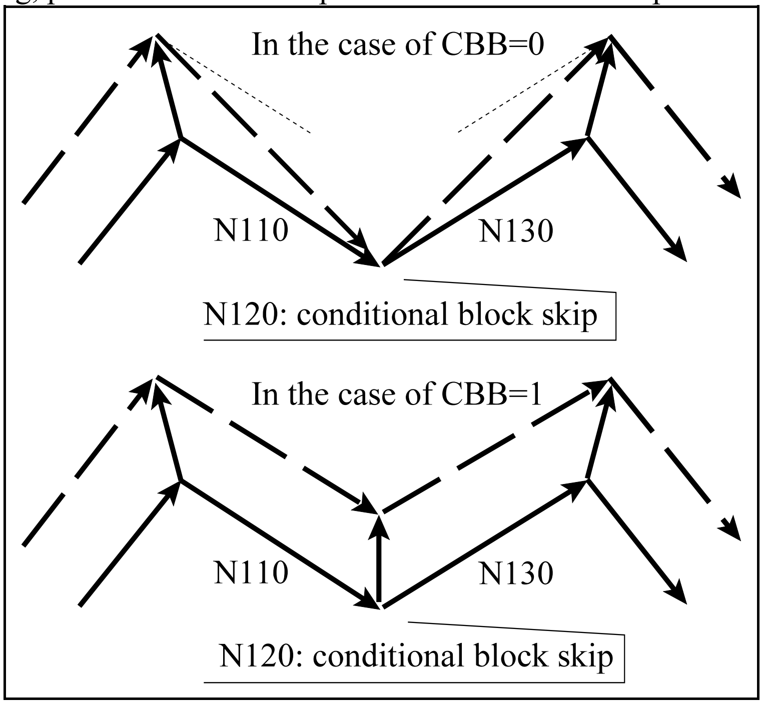


Fig. 15.6.2-9

15.6.3 Cancelling the Tool Nose Radius Compensation. Leaving the Contour

The command G40 cancels calculation of tool nose radius compensation. The command G40 can be issued by linear interpolation only. If G40 is programmed in circular block, the control will send the error message '2043 G40 in circle interpolation'.

The basic cases of cancelling the tool nose radius compensation:

(G42)	(G42)
G01 X_ Z_	G02 X_ Z_ R_
G40 X_ Z_	G40 G1 X_ Z_

Leaving an inner corner: 180°<α<360°

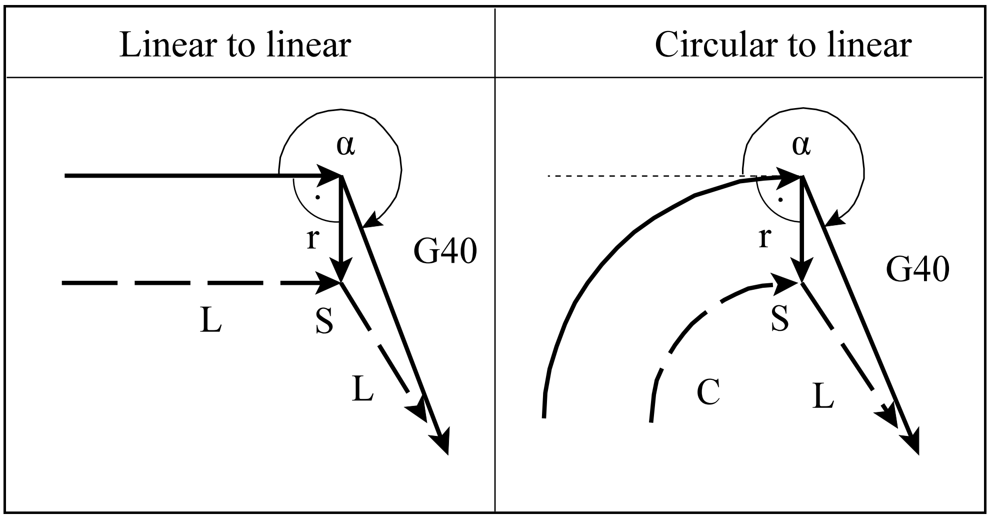


Fig. 15.6.3-1

Leaving an outer corner at an obtuse angle: 90°≤α≤180°

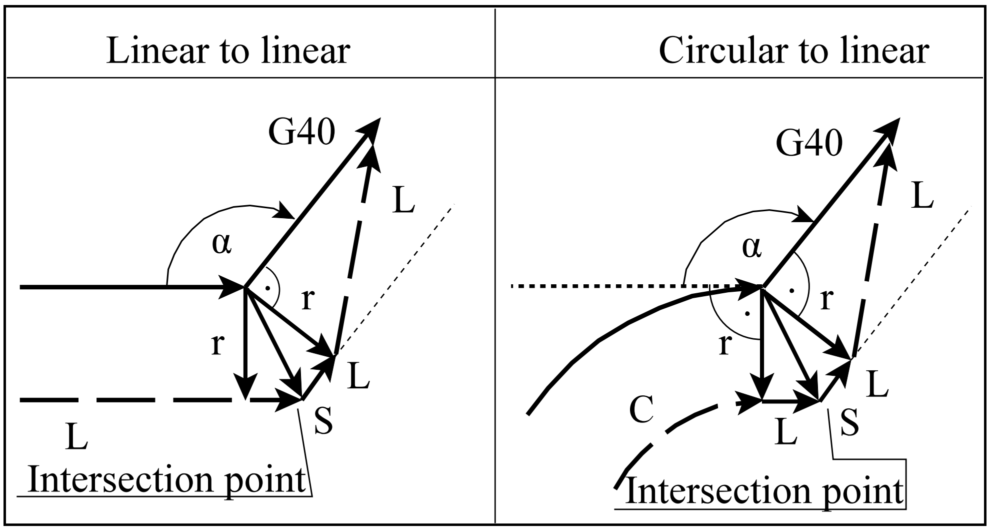


Fig. 15.6.3-2

Leaving an outer corner at an acute angle: 0°≤α<90°

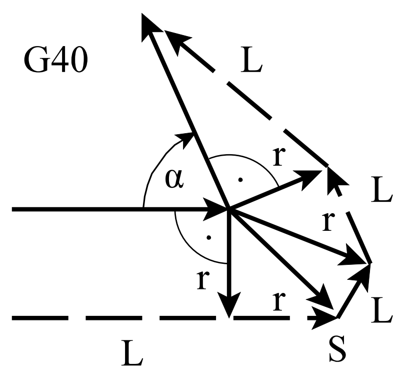
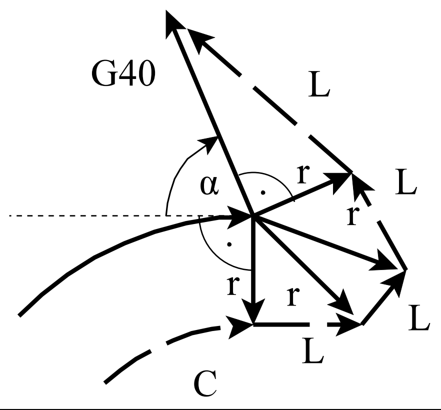
Linear to linear	Circular to linear
	

Fig. 15.6.3-3

An example:
The program below is an example for correct cancelling the tool nose radius compensation at the code of imaginary tool tip Q3.
The tool nose radius compensation is cancelled by the block N180. The control positions to the end of the block 170 with the tool nose radius, the imaginary nose of the tool overhangs the endpoint, and then, at the end of the block N180, the imaginary tool nose will be at the endpoint coordinate:

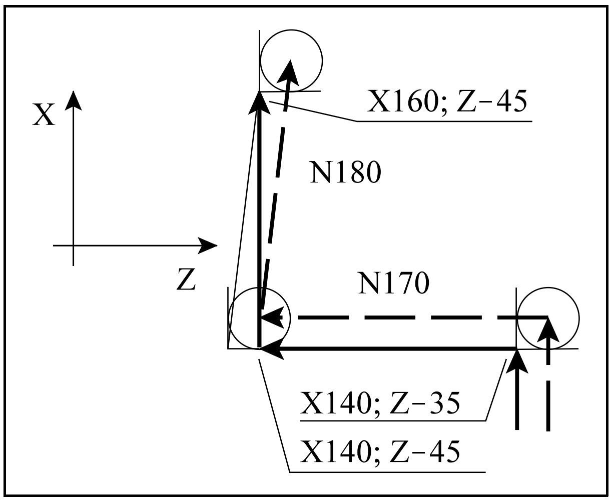


Fig. 15.6.3-4

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
N110 G1 G42 Z0
N120 X100
N130 X110 Z-5
N140 Z-25
N150 G2 X130 Z-35 R10
N160 G1 X140
N170 Z-45
N180 G40 X160
N190 G0 X200 Z50
...
```

Cancelling the tool nose radius compensation without programming of motion:
If ***tool nose radius compensation is cancelled*** (G40) in a block in which there are ***none of the addresses of the axes belonging to the selected plane*** (as in the example below, neither the address X nor the address Z is given in the block N190), motion in the block N190 will not be executed. First, in the block N180, the control guides the tool radius to the endpoint , and then, retracting the tool, it moves the distance of tool nose radius.
Correction can be done by merging the blocks N190 and N200.

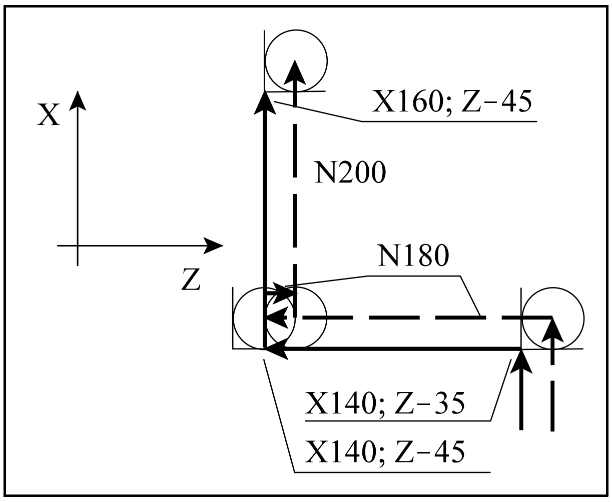


Fig. 15.6.3-5

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
N110 G1 G42
N120 Z0
N130 X100
N140 X110 Z-5
N150 Z-25
N160 G2 X130 Z-35 R10
N170 G1 X140
N180 Z-45
N190 G40
N200 X160
N210 G0 X200 Z50
...
```

If ***tool nose radius compensation is cancelled*** (G40) in a block in which ***there is no displacement along none of the axes belonging to the selected plane*** (as in the example below, in the block N190 the address X is filled, but because of the incremental 0 belonging to it, there is no motion), the situation will be the same as in the case above, when axis address was not programmed.

```
G54 G18
...
N50 G0 X200 Z50
N60 G92 S3000
N70 G96 S600
N80 T101 M3
N90 X160 Z3
N100 X90
N110 G1 G42 G91 Z0
N120 G90 Z0
N130 X100
N140 X110 Z-5
N150 Z-25
N160 G2 X130 Z-35 R10
```

```
N170 G1 X140
N180 Z-45
N190 G91 G40 X0
N200 G90 X160
N210 G0 X200 Z50
...
```

Cancelling tool nose radius compensation with calculation of intersection point

If, in the block carrying out cancelling (G40), a value is assigned to I, J and K, but only to those that are in the selected plane (for example, to I and K in the case of G18), the control will move to the intersection point defined by the previous block and by I, J and K. The value of I, J and K is always incremental, and the vector defined by them points from the endpoint of the previous block.
It is a useful thing in the case of leaving an inner corner, for example.

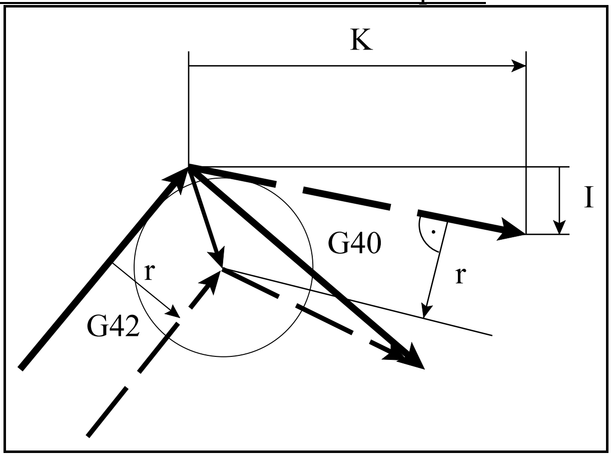


Fig. 15.6.3-6

An example:
A hexagon with side length of 100 mm is to be milled internally. The tool has to be left from the corner point with the coordinates of X100 Y0.The block N120 of the program below leaves from the contour using calculation of intersection point, with specification of I and J. The coordinates of I and J were calculated on the basis of the displacement executed in the block N60 of the program.

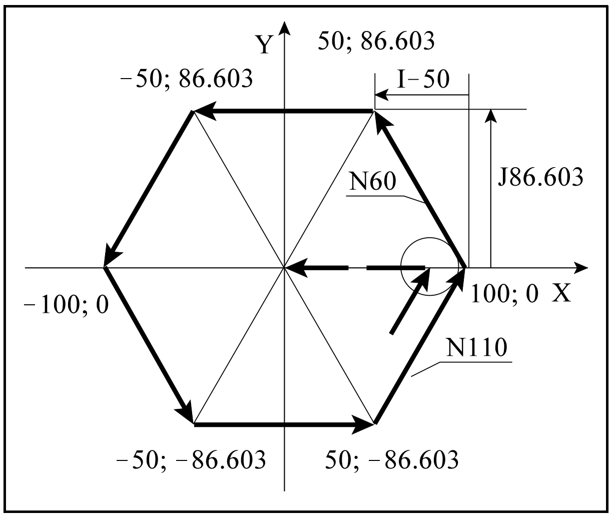


Fig. 15.6.3-7

```
N10 G54 G17 M3 S300
N20 G0 X0 Y0 Z100
N30 G43 Z5 H1
N40 G1 Z-5 F1000
N50 G41 G0 X100 Y0 I50 J86.603 D1
N60 G1 X50 Y86.603
N70 X-50
N80 X-100 Y0
N90 X-50 Y-86.603
N100 X50
N110 X100 Y0
N120 G40 G0 X0 I-50 J86.603
N130 M30
```

Cases of leaving using intersection point

In the case of specification I, J and K, the control always calculates intersection point regardless of whether inner or outer corner is to be machined.

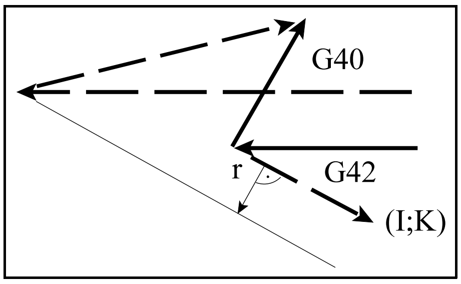


Fig. 15.6.3-8

If the control does not find intersection point, it will position to the endpoint of the previous block at right angles.

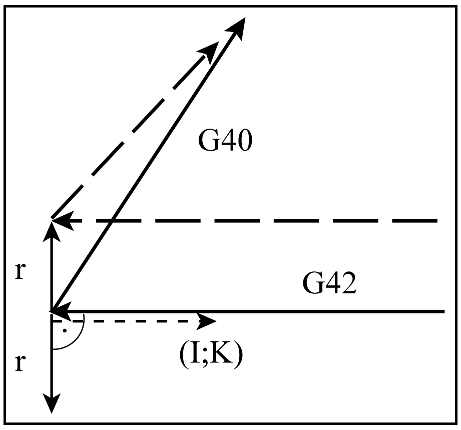


Fig. 15.6.3-9

15.6.4 Reversal in Calculation of Tool Nose Radius Compensation

The following table shows the direction of calculation of tool nose radius compensation, i.e. the direction of tracking the contour:

	Tool radius compensation: **positive**	Tool radius compensation: **negative**
G41	from the left	from the right
G42	from the right	from the left

The direction of tracking the contour can also be reversed when the calculation of tool nose radius compensation is started up. It can be carried out by programming G41 or G42. When the direction of tracking the contour is being changed, the control will not check whether the position is 'outside' or 'inside', but it will always calculate intersection point first. In the examples illustrated below, positive tool nose radius and change over from G42 to G41 are assumed:

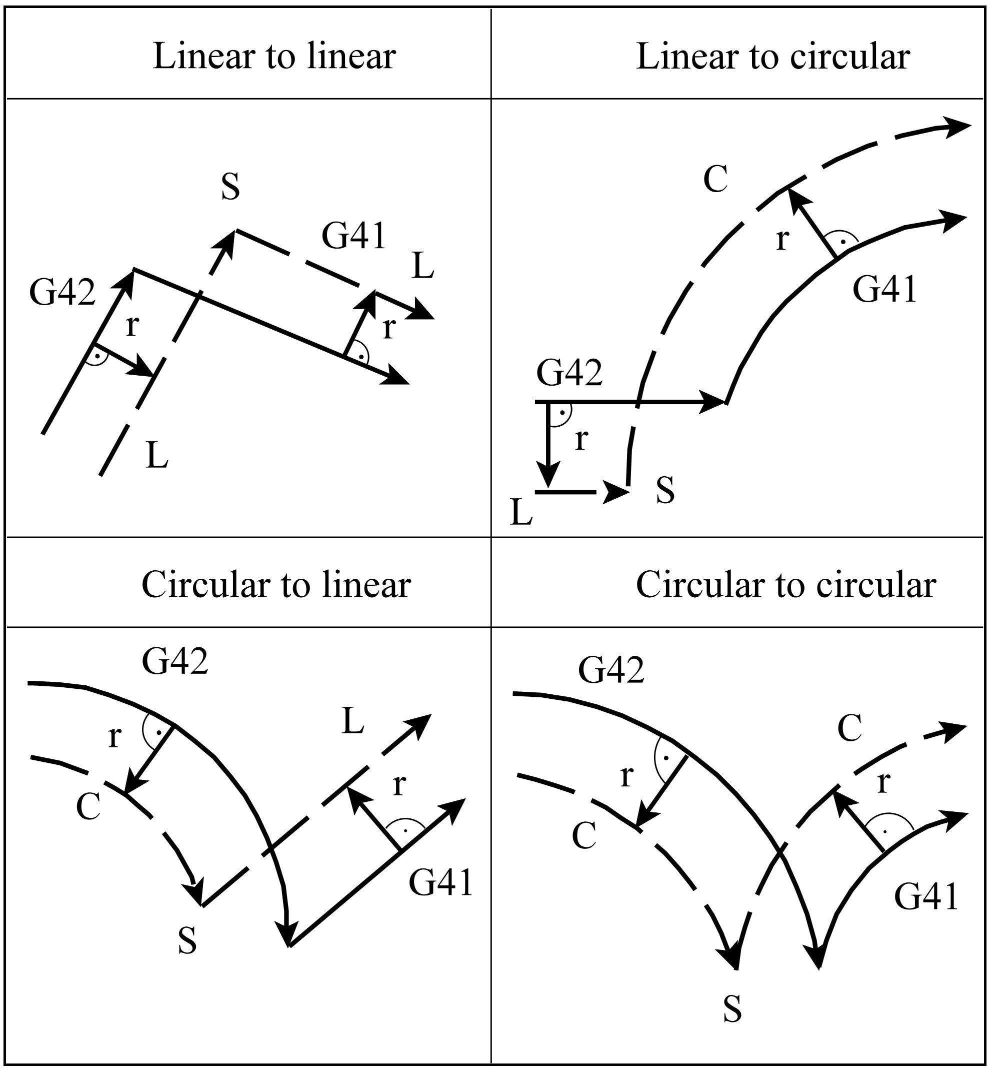


Fig. 15.6.4-1

An example:
Positioning to a circle with radius of R50 is executed in the blocks N30 and N40, along a straight line from the left, using G41. Then, the circle is tracked in the block N50, from the right, using G42. Leaving is also executed along a straight line from the left, using G41.

```
...
N10 G17 G0 X100 Y0
N20 T101 M3 S500
N30 G41 X80
N40 G1 X50
N50 G42 G3 I-50
N60 G41 G1 X80
N70 G40 X100
...
```

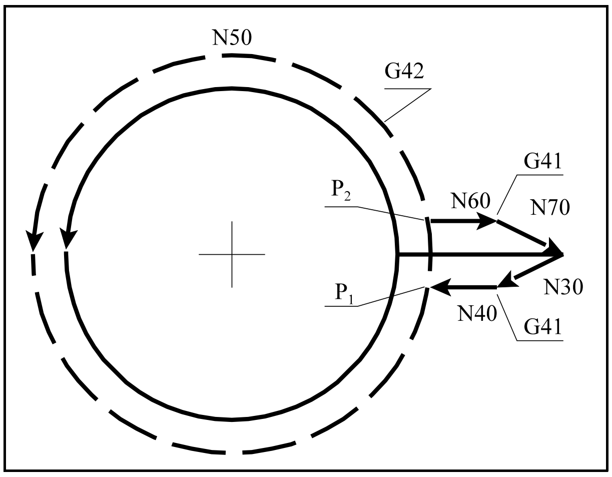


Fig. 15.6.4-2

Programming reversal along full circle

Programming full circle with reversal G41-G42, several cases can occur, when the path covered by the tool will be longer than the length of the full circle.
The example above illustrates this case.
The center point of the tool covers a full circle from the point P1 up to the point P1, and then a circle arc from the point P1 up to the point P2.

There is no intersection point in the case of linear - linear transition

If, in the case of linear - linear transition, no intersection point is resulted, the path of the tool will be as it is illustrated in the figure. According to the figure, reversal will occur from G42 to G41.
In this case, in the endpoint of the block preceding the reversal, the endpoint will be the vector with the length of tool radius which is put perpendicularly to the starting point of the succeeding straight line.

The upper figure shows the case when the direction of the path does not change, the lower figure illustrates the case of the path reversal of 180°.

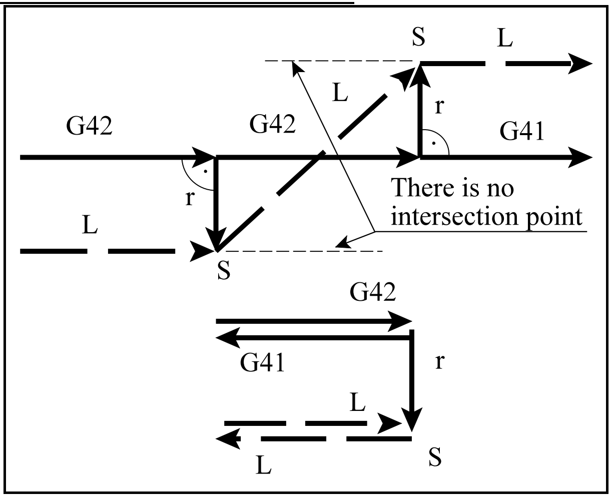


Fig. 15.6.4-3

There is no intersection point in the case of linear - circular transition

If, in the case of linear - circular transition, no intersection point is resulted, the path of the tool will be as it is illustrated in the figure. According to the figure, reversal will occur from G41 to G42.

In this case, in the endpoint of the straight line preceding the reversal, the endpoint will be the vector with the length of tool radius which is put perpendicularly to the starting point of the succeeding straight line.

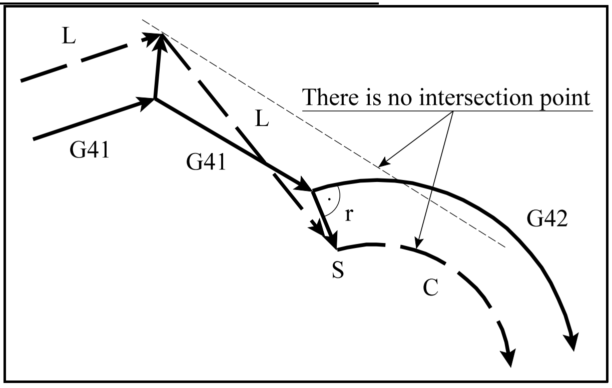


Fig. 15.6.4-4

There is no intersection point in the case of circular - linear or circular - circular transition

If, in the case of circular - linear or circular - circular transition, no intersection point is resulted, the control will interconnect the endpoint of the compensation vector resulted in the starting point of the first circular block and the endpoint of the compensation vector put perpendicularly to the starting point of the second block using an uncorrected programmed circle arc with radius R. In this case, the center point of the interconnecting circle arc will not coincide with the center point of the programmed circle arc. If reversal cannot be executed even with this change over of center point of circle, the control will send an error message.

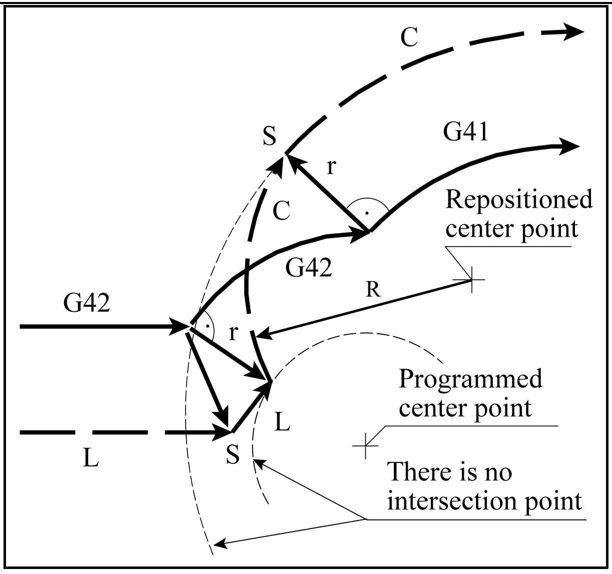


Fig. 15.6.4-5

15.6.5 Programming Vector Preservation (G38)

Due to the command

G38 v,

in started up state of calculation of planar tool nose radius compensation, the control ***preserves the last compensation vector between the preceding block and the block of G38, and validates it at the end of the block of G38***, irrespective of transition between the block of G38 and the succeeding block.

The code is a one-shot code, i.e. it is not a modal one. If it is necessary to preserve the vector in several consecutive blocks, the code G38 has to be programmed again.

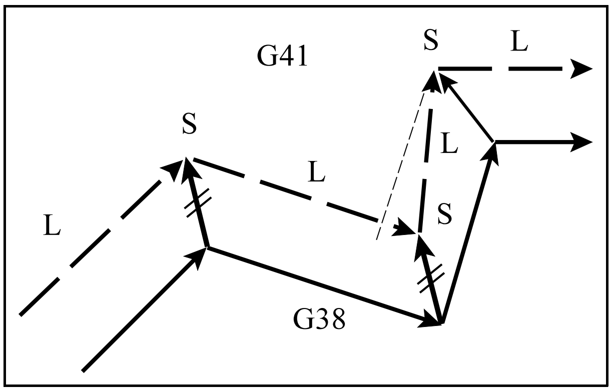


Fig. 15.6.5-1

An example:

Grooving is to be programmed without cancelling the contour tracking:

```
...G18 G42 G91...
N110 G1 Z40
N120 G38 Z50
N130 G38 X140
N140 G38 X-140
N150 Z60
...
```

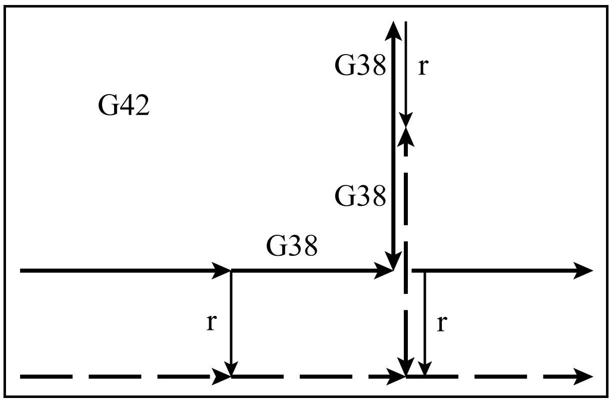


Fig. 15.6.5-2

15.6.6 Programming Corner Arc (G39)

When programming a block

G39 (I J K)

in the started up state of calculation of planar tool nose radius compensation, it can be reached, that in the case of going around an outer corner the control will not calculate intersection point automatically or will not insert straight line segments for going around; instead, the center point of the tool will move along a circle arc with radius equal to the tool nose radius.

The control ***inserts a circle arc with radius equal to the tool nose radius***; the direction of the circle arc is G02 in the state G41, and G03 in the state G42.

The starting point of the circle arc is defined by the vector perpendicular to the endpoint of the preceding block and length of which is equal to the tool nose radius, while the endpoint of the circle arc is defined by the vector perpendicular to the starting point of the succeeding block and length of which is equal to the tool nose radius. ***G39 must be programmed in separate block***:

```
...G18 G91 G41...
N110 G1 Z100
N120 G39
N130 G3 X-160 Z80 K80
...
```

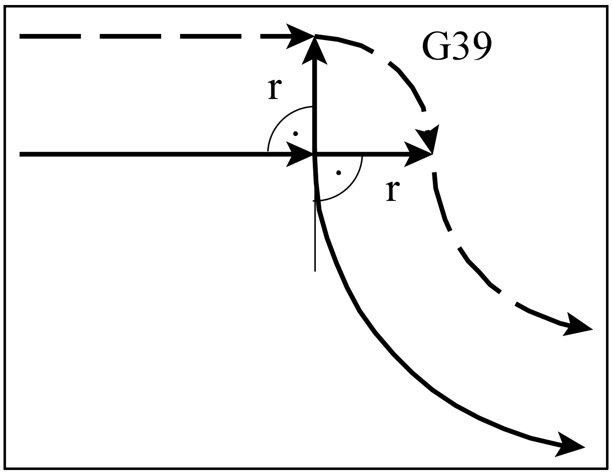


Fig. 15.6.6-1

If I, J or K is programmed in the block of G39 in accordance with the selected plane, the endpoint of the circle arc will be defined by the vector which is perpendicular in the endpoint of the preceding block to the vector defined by I, J or K and length of which is equal to tool nose radius:

```
...G18 G91 G41...
N110 G1 Z100
N120 G39 I-60 K50
N130 G40 X60 Z110
...
```

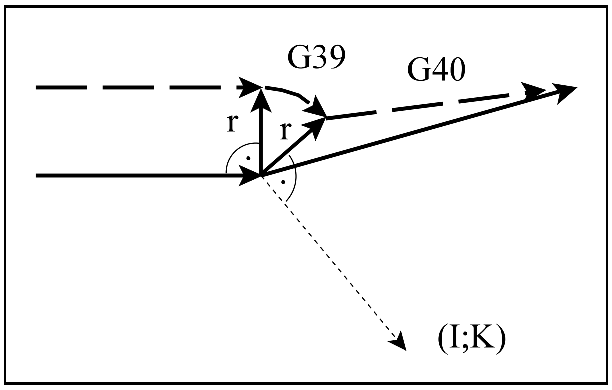


Fig. 15.6.6-2

Commands on mirroring or rotation set previously are valid for the vector defined by I, J or K. Direction, certainly, is not affected by scaling command.

In the block of G39, no motion command of any kind can be programmed.

An example:
A triangular slot is to be milled with fillet at the corners.
In order that the contour will be closed, J is used for giving G39 in the block N110. Data of I and J are specified from displacement of the block N60.

```
...
N20 G0 G17 G40 G54
N30 X-30 Y150 Z5 M3 S300
N40 G42 X0 Y86.603 D1
N50 Z-2
N60 G1 X-50 Y0 F1000
N70 G39
N80 X50
N90 G39
N100 X0 Y86.603
N110 G39 I-50 J-86.603
N120 Z5
N130 X-40
N140 G40 Y120
...
```

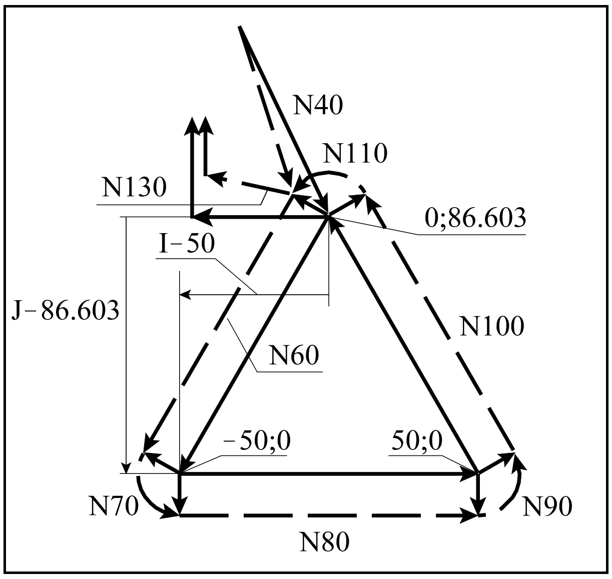


Fig. 15.6.6-3

15.6.7 Troubles in Tracking the Contour. Interference Check

During execution of contour tracking, it occurs in many cases that the tool path will be opposite of the programmed path. In this case, ***the tool can cut into the workpiece but that is not the intention of the programmer***. This phenomenon is called ***trouble in tracking the contour*** or ***interference***.
In the case illustrated in the figure, after calculation of the intersection points, during execution of the block N2, a tool path (dash line) opposite to the programmed path (continuous line) will be generated. The tool will cut into the workpiece.

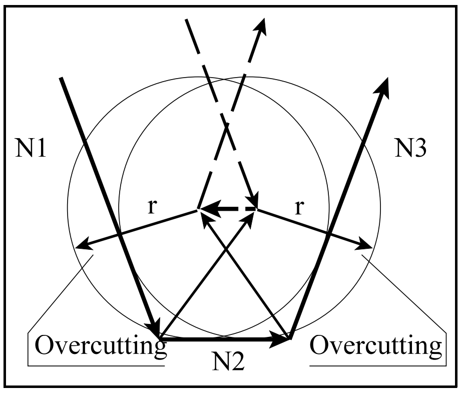


Fig. 15.6.7-1

In order to avoid such cases, the control executes interference check at the bit position #0 IEN=1 of the parameter N1403 Interference. In this case, ***the control checks whether the angle φ between the programmed displacement and the displacement corrected by the radius compensation satisfies the condition -90°≤φ≤+90°***.
In other words, the controls checks whether the compensated displacement vector has a component opposite to the programmed displacement vector.

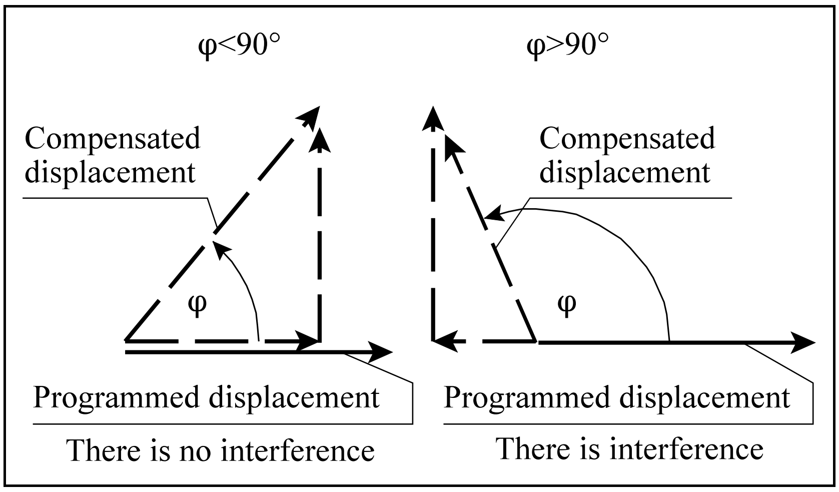


Fig. 15.6.7-2

If the control detects interference, ***it will either indicate the error or try to correct the error***, according to the bit position #1 AAL of the parameter N1403 Interference.
If ***AAL=1***, the control ***will always send the error message '2049 Interference alarm'*** at the end of the block preceding the block which causes the interference.
If ***AAL=0***, the control ***will try to correct the error automatically***, and it will send an error message in the only case, if the automatic correction gives no result.

For monitoring interference and automatic correction, ***the control normally reads 3 blocks in ahead***. ***If the value of the parameter N1404 BK No. Interf is greater than 0, the control will read in blocks quantity of which is a parameter value 3+*** and will execute interference check.

Automatic correction of interference

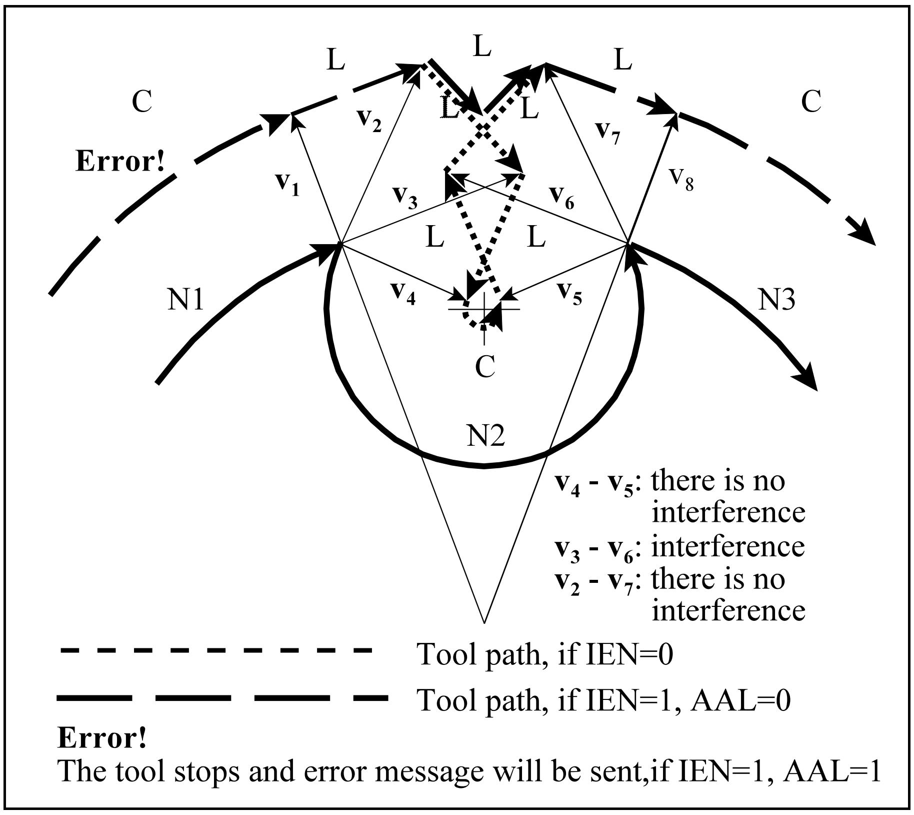

L
L
L
C
C
Error!
v1
v2
L
L
v7
v8
v3
v6
L
L
N1
v4
v5
N3
C
N2
v4 - v5: there is no interference
v3 - v6: interference
v2 - v7: there is no interference
Tool path, if IEN=0
Tool path, if IEN=1, AAL=0
Error!
The tool stops and error message will be sent,if IEN=1, AAL=1

Fig. 15.6.7-3

If AAL=0, the control will not indicate error, but it will try to correct the contour automatically in order to avoid overcuttings. The course of the correction is as follows:

The tool nose radius compensation (G41) is started up in the blocks N1, N2 and N3.

The calculated compensation vectors between the blocks N1 and N2: $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ és $\mathbf{v}_4$.

The calculated compensation vectors between the blocks N2 and N3: $\mathbf{v}_5$, $\mathbf{v}_6$, $\mathbf{v}_7$ és $\mathbf{v}_8$.

If there is interference between the vectors $\mathbf{v}_4$ and $\mathbf{v}_5$ (a displacement opposite to the displacement of N2 is generated), the control will calculate intersection point between the straight line defined by the vectors $\mathbf{v}_2$ and $\mathbf{v}_3$ and the straight line defined by the vectors $\mathbf{v}_6$ and $\mathbf{v}_7$, and it will skip the circle arc N2.

If there is interference between the vectors $\mathbf{v}_3$ and $\mathbf{v}_6$ (a displacement opposite to the displacement of N2 is generated), the control will calculate intersection point between the straight line defined by the vectors $\mathbf{v}_2$ and $\mathbf{v}_3$ and the straight line defined by the vectors $\mathbf{v}_6$ and $\mathbf{v}_7$, and it will skip motions between them (the figure above shows this case).

If there is interference between the vectors $\mathbf{v}_2$ and $\mathbf{v}_7$ (a displacement opposite to the displacement of N2 is generated), the control will calculate intersection point between the straight line defined by the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ and the straight line defined by the vectors $\mathbf{v}_7$ and $\mathbf{v}_8$, and it will skip motions between them.

If there is interference between the vectors $\mathbf{v}_1$ and $\mathbf{v}_8$, the control will try to calculate an intersection point between the blocks N1 and N3.

It is evident from the example above, that execution of the block N1 will start only after the interference check for the block N2 is carried out by the control. For this, however, the control had to read the block N3 into the buffer too, and to calculate the compensation vectors at the transition N2 - N3.

If, in the block N2, in the case of AAL=0, the control cannot eliminate interference by calculation of an intersection point between the compassing segments, it will try to calculate an intersection point between the blocks N1 and N3. If there is an intersection point, the control will go on, otherwise it will send the error message '2049 Interference alarm'.

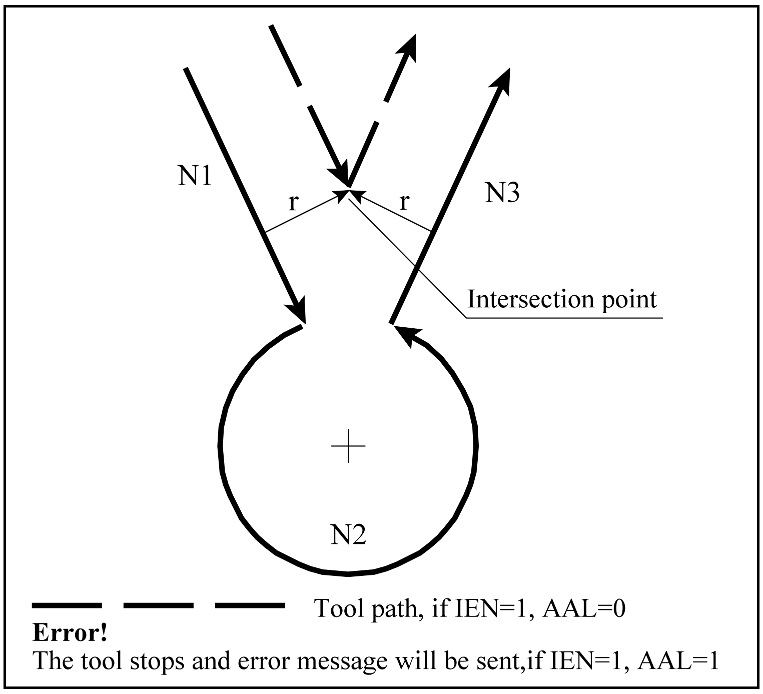

N1
r
r
N3
Intersection point
N2
Tool path, if IEN=1, AAL=0
Error!
The tool stops and error message will be sent,if IEN=1, AAL=1

Fig. 15.6.7-4

Monitoring the interference several blocks in advance

In the examples above, 3 blocks are checked by the controller. Namely, the transition between N1 and N2, and the transition between N2 and N3. It is adequate to the parameter value N1404 BK No. Interf=0.

The maximum value of the parameter N1404 BK No. Interf can be 8. In this case, the control executes the checks above

for the blocks (displacements) between N1 and N2, and, N2 and N3;

for the blocks (displacements) between N1 and N2, and, N3 and N4;

...

for the blocks (displacements) between N1 and N2, and, N [BK No. Interf+2] and N [BK No. Interf+3]. Then, depending on the position of the parameter AAL, in the block N1 the control will indicate error, or it will try carry out correction.

It can be used in the case when the tool enters into a hollow and it has to be checked whether there is enough room within for the tool with its diameter.

If the hollow illustrated in the figure is to be checked by the control, the value N1404 BK No. Interf=7 will have to be set.

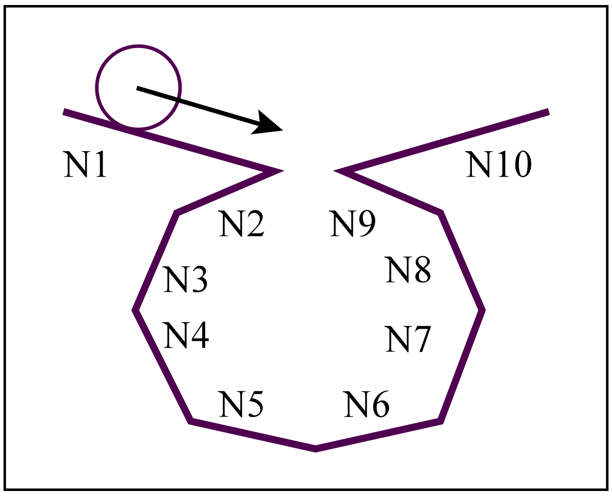


Fig. 15.6.7-5

Typical interferences

There are described below some typical cases when the control detects interference.

Producing a ***step with height smaller than the radius of the tool***.
In the case of IEN=0: the control cuts into the workpiece.
In the case of IEN=1:
if AAL=0, the control will avoid cutting into the workpiece by calculating an intersection point between the blocks N1 and N3;
if AAL=1, the control will send the error message '2049 Interference alarm', because it would cut into the workpiece.

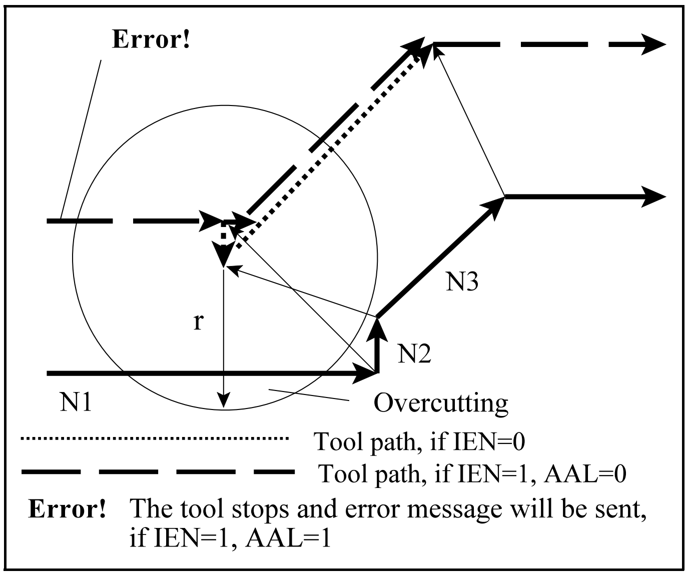

Error!
N3
r
N2
N1
Overcutting
Tool path, if IEN=0
Tool path, if IEN=1, AAL=0
Error! The tool stops and error message will be sent, if IEN=1, AAL=1

Fig. 15.6.7-6

Producing an ***inner corner with a radius smaller than the radius of the tool***.
In the case of AAL=1, the control will send the error message '2049 Interference alarm'.
In the case of AAL=0, the control, skipping the circle and calculating an intersection point between the two straight lines, will correct the error.

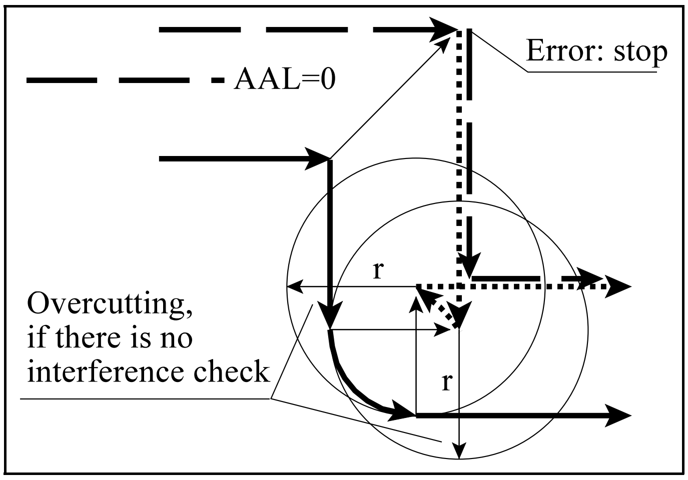


Fig. 15.6.7-7

Producing a ***step with height smaller than the radius of the tool, along a circle arc.***
If AAL=0, the control will calculate an intersection point between the straight line L1, and the straight line interconnecting the vectors $\mathbf{v}_1$ és a $\mathbf{v}_3$, in order to avoid the cutting in.
If AAL=1, the control will send the error message '2049 Interference alarm' and it will stop in the previous block.

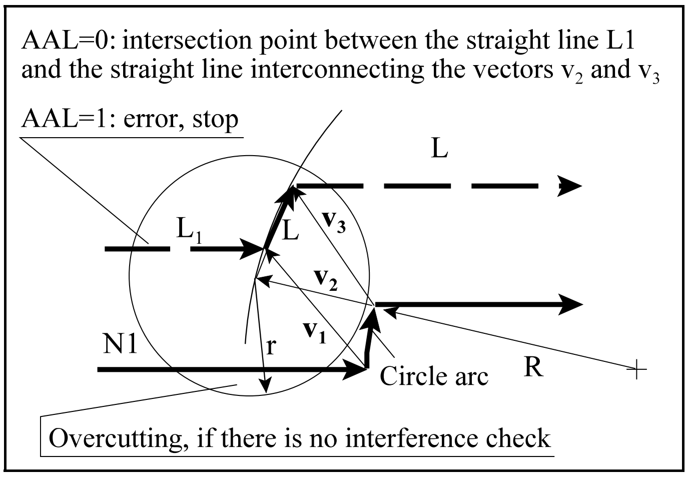

AAL=0: intersection point between the straight line L1 and the straight line interconnecting the vectors v_2 and v_3
AAL=1: error, stop
L
L_1
L
v_3
v_2
N1
v_1
r
Circle arc
R
Overcutting, if there is no interference check

Fig. 15.6.7-8

In the case of the path shown in the figure, in the block N2 the displacement along the compensated path is opposite to the programmed path.
The control cannot correct if the value of the parameter N1404 BK No. Interf is 0, because there is no intersection point between the blocks N1 and N3, so it will indicate error in both cases of AAL=0 and AAL=1.
If the value of the parameter ***N1404 BK No. Interf is greater than 0,*** the control will continue reading the blocks, therefore it ***will correct interference*** by creating an intersection point between the blocks N1 and N4, skipping the blocks N2 and N3.

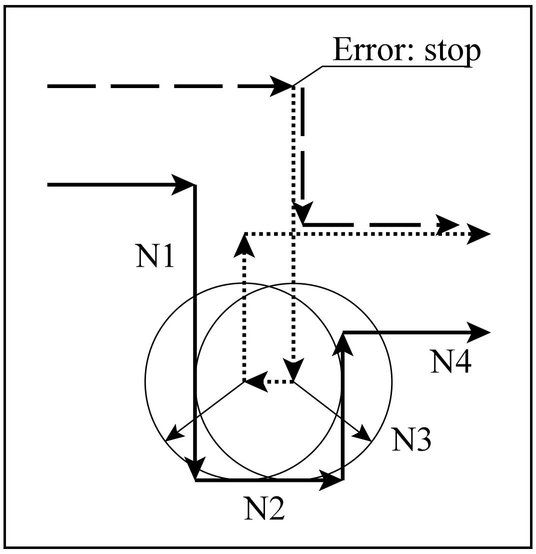


Fig. 15.6.7-9

In the case of the path shown in the figure, in the block N2 the displacement along the compensated path is opposite to the programmed path.
In the case of AAL=1, the control will indicate error at the starting point of the block N1.
In the case of AAL=0, the control will calculate an intersection point between the blocks N1 and N3, and moves up to this intersection point. If it proceeded from this intersection point, a motion opposite to the block N3 would be created along the corrected path, therefore the control will indicate error if the value of the parameter N1404 BK No. Interf is 0.

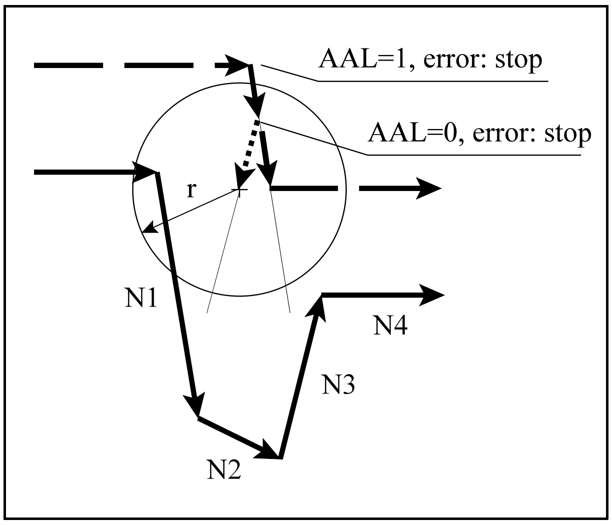


Fig. 15.6.7-10

If the value of the parameter ***N1404 BK No. Interf is greater than 0,*** the control will continue reading the blocks, therefore it ***will correct interference*** by creating an intersection point between the blocks N1 and N4, skipping the blocks N2 and N3.

Indication of interference, but no cutting into the workpiece

There are cases, when the interference check indicates error, but the control would not cut into the workpiece.

If producing a recess with depth smaller than the radius compensation is to be produced, maybe there will not be cutting into the workpiece in reality, as it is shown in the figure, but the control will send the error message '2049 Interference alarm' in the case of AAL=1, because in the block N3 the direction of the displacement along the corrected path is opposite to the programmed one.

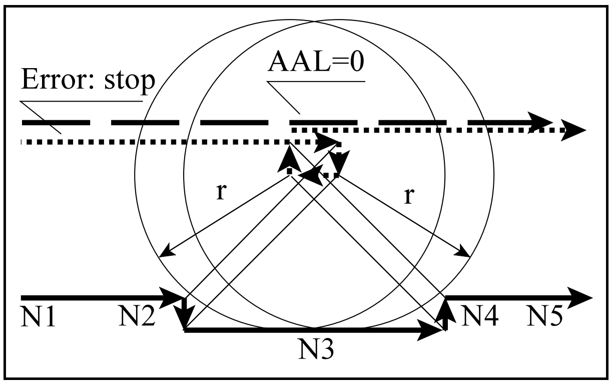


Fig. 15.6.7-11

In the case of AAL=0, the control continues machining, skipping the blocks N2, N3 and N4 and interconnecting the blocks N1 and N5, as it is illustrated in the figure.

In the example shown in the figure, the control also indicate interference, because in the block N3 the displacement along the corrected path is opposite to the programmed one.

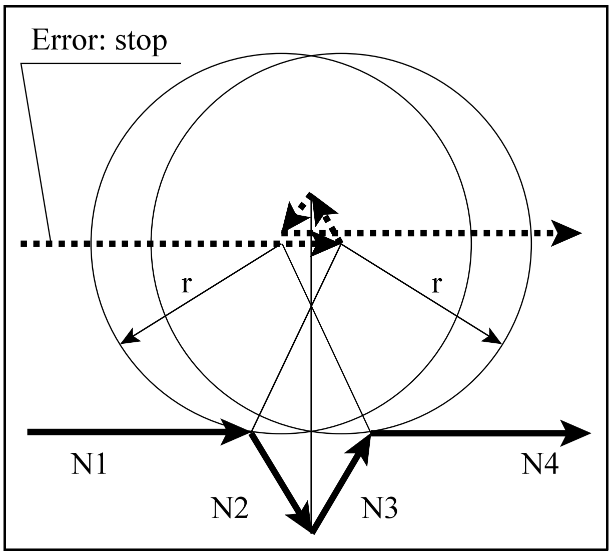


Fig. 15.6.7-12

There is cutting into the workpiece despite the monitoring interference

There are cases from the geometry of the path, when the control cuts into the workpiece, despite the monitoring interface. Some cases are shown below.

In the case illustrated in the figure, in the block N3, the displacement occurs in the direction opposite to the programmed one, and for this reason, the control will indicate interference at the starting point of block N2 and it will stop if the value of the parameter N1404 BK No. Interf is 0. Because of the geometry (G2) of the N4, the control will cut into the workpiece .
If the value of the parameter N1404 BK No. Interf is greater than 0, the control will continue reading the blocks, and for this reason, it will ***correct the interference*** by the intersection point between the blocks N1 and N4, skipping the blocks N2 and N3 in the case of AAL=0, otherwise it will indicate error at the beginning of the blockN1.

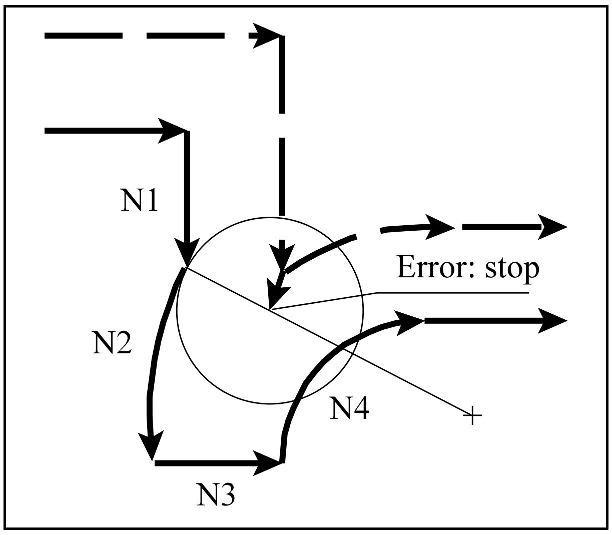


Fig. 15.6.7-13

In the case shown in the figure, no displacement occurs in the direction opposite to the programmed one in none of the blocks N1, N2 and N3, and for this reason, the control will not indicate interference if the value of the parameter N1404 BK No. Interf is 0, but even so, it will cut into the workpiece because of the geometry of the path.
If the value of the parameter N1404 BK No. Interf is greater than 0, the control will continue reading the blocks, and for this reason, it will ***correct the interference*** by the intersection point between the blocks N1 and N4, skipping the blocks N2 and N3 in the case of AAL=0, otherwise it will indicate error.

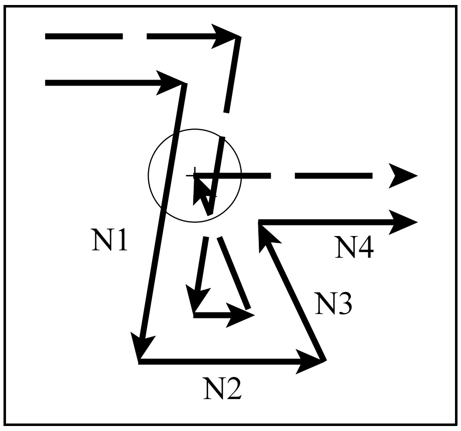


Fig. 15.6.7-14

In the case of the path illustrated in the figure, in the block N2, the displacement along the corrected path would be opposite to the programmed one.
In the case of AAL=1, the control indicates error at the starting point of the block N1.
If AAL=0, the control will calculate an intersection point between the blocks N1 and N3 and it will continue the motion up to this intersection point. If the control continued the motion from this intersection point, a motion opposite to the block N3 would occur, and for this reason the control indicates error.
However, at the end of the block, cutting into the workpiece will already occur if the value of the parameter N1404 BK No. Interf is 0.

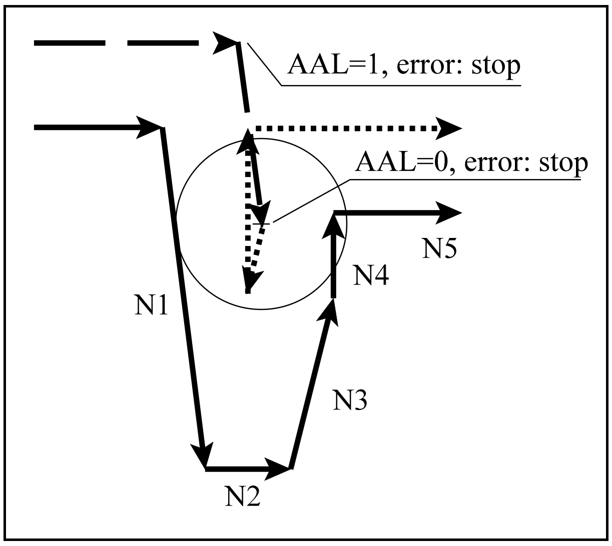


Fig. 15.6.7-15

If the value of the parameter N1404 BK No. Interf is 1, the control will indicate interference.
If the value of the parameter N1404 BK No. Interf is 2, the control will correct the interference by calculation of an intersection point between the blocks N1 and N5.

16 Special Transformations

16.1 Rotating a Shape Around a Given Point (G68, G69)

By the use of the command

G68 p q R

a programmed shape ***can be rotated in the plane selected by G17, G18 and G19***.

The ***coordinates of the center of rotation are*** specified at the addresses

p and ***q***.

The only values written at the coordinates p and q of the selected plane are interpreted by the control.

The entered ***coordinate data p and q*** will be interpreted by the control ***in the orthogonal coordinate system***, even if polar coordinate data specification is valid.

The coordinates p and q can be specified as ***either absolute or incremental*** data by the use of G90, G91 or the operator I. The incremental data has to be interpreted from the last programmed axis position (not from the rotated one).

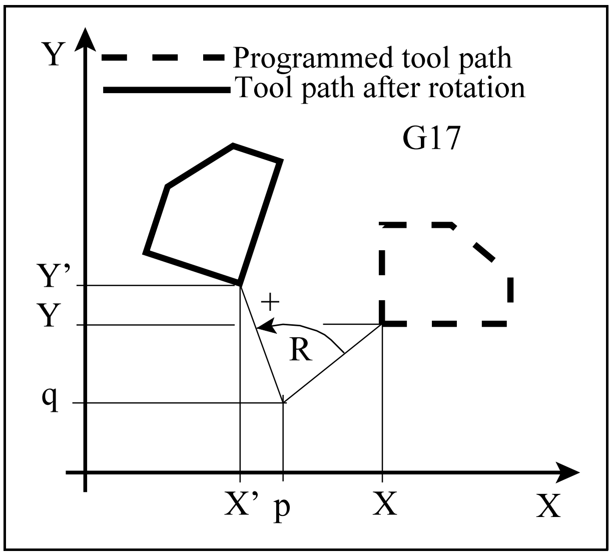


Fig. 16.1-1

If value is not assigned to one of the p and the q or both of them, the instantaneous axis position will be interpreted as the center of rotation.

The equation of rotation of a shape in the plane XY, in the case of G17 is the following:

$$X' = (X - p)\cos R - (Y - q)\sin R + p$$

$$Y' = (X - p)\sin R + (Y - q)\cos R + q$$

where: p, q: the center of rotation;
R: the angle of rotation;
X, Y : the coordinates of the programmed point;
X', Y': the coordinates of the rotated point.

It is the address

R,

at which the ***angle of rotation*** is specified. A ***positive*** or ***negative*** value entered at the address represents ***counter-clockwise*** or ***clockwise*** direction, respectively, to be interpreted in the selected plane.

The value assigned to the R can be ***either absolute or incremental***. If incremental angel of rotation is specified, the value of R will be added to the angles of rotation programmed previously.

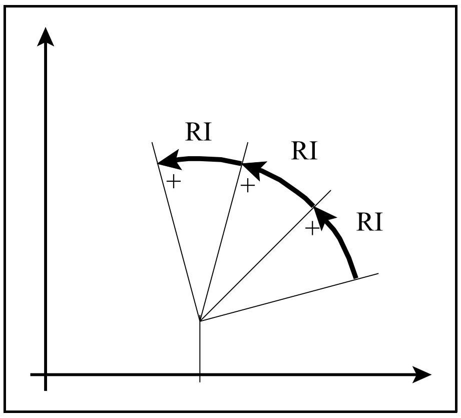


Fig. 16.1-2

The command

G69

cancels rotation. It deletes the coordinates of of the center of rotation and the angle of rotation, too. This command can also accompany other commands, too.

An example:

Three squares positioned in relation to one another at the angle of 120° is to be milled into the face of a workpiece, using polar interpolation. The square is described by the subprogram O0001. The workpiece will shape by calling the subprogram three times with incremental rotation of 120° at the end of the subprogram.
In the picture, the order of execution is marked with the numbers 1, 2 and 3.

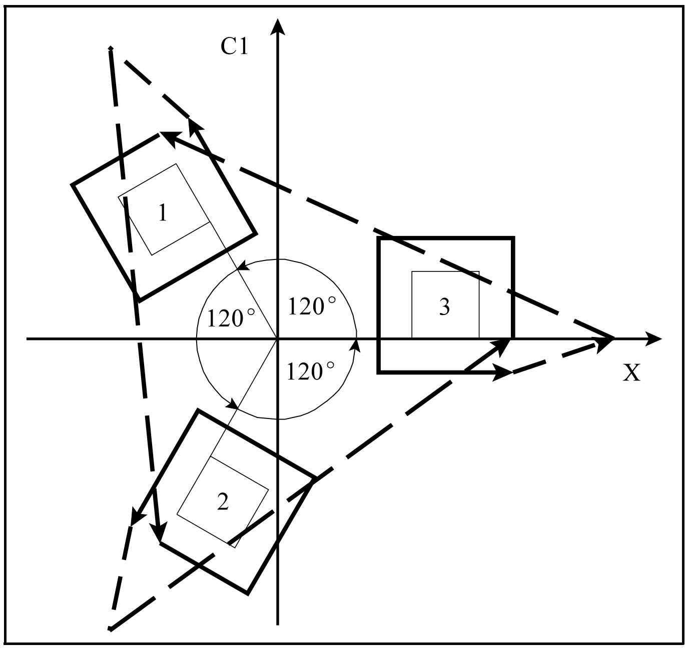


Fig. 16.1-3

Main program:

```
G17 G59 G94 G0 X100 Z20 C1=0
M3 S2=500
G12.1
G68 X0 C1=0 R120
M98 P1 L3
G69
G13.1
```

Subprogram:

```
O0001
G42 G0 X60 C1=0
G1 Z-2
C1=10
X40
C1=0
X70
G40 G0 X100 C1=0
Z20
G68 X0 C1=0 RI120
M99
```

16.2 Scaling a Shape in Relation to a Given Point (G50, G51)

Scaling can be activated by the code G51. Two ways are available for scaling specification: one of them is, when a common magnification rate is valid for each axis; the other is, when different rates of magnification are applied to the axes X, Y and Z.
Both ways of specification can be cancelled by the command G50.

Scaling by the use of a magnification rate valid for each axis (P)

Using the command

G51 v P,

a programmed shape can be ***reduced or magnified***.

At the coordinates

v ,

the ***position of the center point of scaling*** can be specified. The axes that can be used are the axes X, Y and Z, and the parallel axes.

The entered ***coordinate data v*** will be interpreted by the control ***in the orthogonal coordinate system***, even if polar coordinate data specification is valid.

The coordinates v of the center point of scaling can be specified as ***either absolute or incremental*** data by the use of G90, G91 or the operator I.

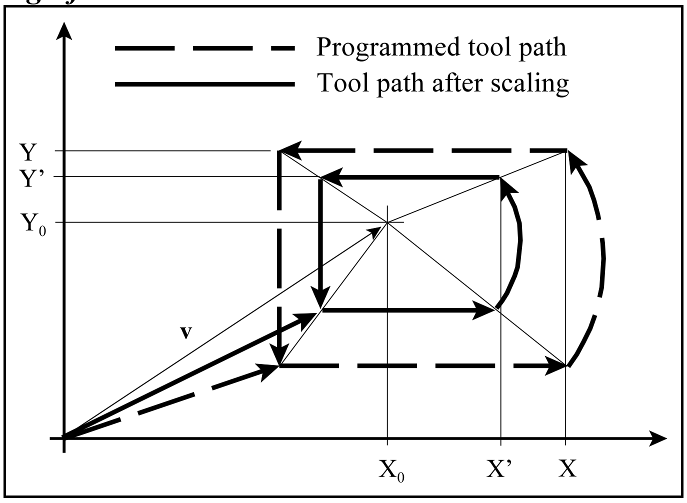


Fig. 16.2-1

If value is not assigned to one of the axis addresses or each of them, the instantaneous axis position will be interpreted as the center of scaling.

It is the address

P,

at which the ***magnification rate*** is specified.

There will be reduction if P<1, and if P>1, magnification will occur.

The equation of the scaling of a shape in the space XYZ is the following:

$$X' = (X - X_0)P + X_0$$

$$Y' = (Y - Y_0)P + Y_0$$

$$Z' = (Z - Z_0)P + Z_0$$

where:	X_0, Y_0, Z_0: the coordinates v of the center of scaling;
	P: the magnification rate of scaling;
	X, Y, Z: the coordinates of the programmed point;
	X', Y', Z': the coordinates after scaling.

The command

G50

cancels the scaling.

☞ ***Note:***

The magnification rate of scaling (P) does not affect the value of radius compensation applied.

An example:
If the zero point of the workpiece in the direction Z is on the contact surface of the chuck, it will be easy to turn smaller or bigger workpieces by the use of G51.

```
G90 G0 X100 Z120
G51 X0 Z0 P0.5
G1 X0 Z100 F150
X80
Z0
G50
G0 X100 Z120
```

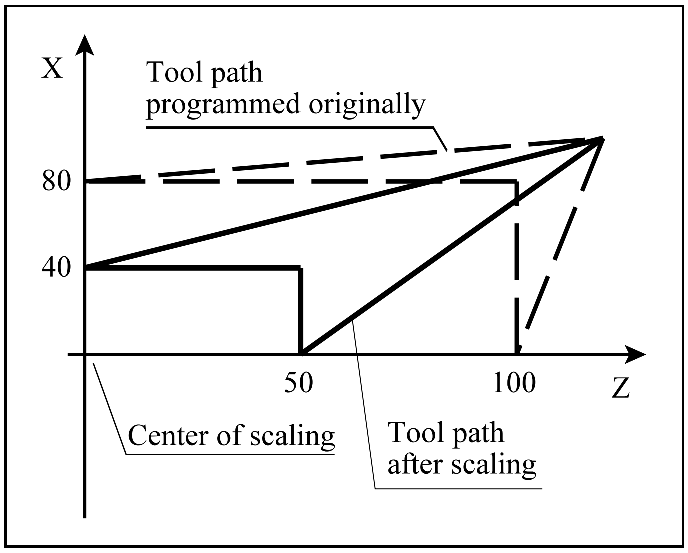


Fig. 16.2-2

<u>Scaling by the use of a scaling rate different for each of the axes X, Y and Z (I J K)</u>

Using the command

G51 X Y Z I J K

a programmed shape can be ***reduced or magnified***.

At the coordinates

X Y Z

the ***position of the center point of scaling*** can be specified. The axes that can be used are the main axes X, Y and Z.

The entered ***coordinate data v*** will be interpreted by the control ***in the orthogonal coordinate system***, even if polar coordinate data specification is valid.

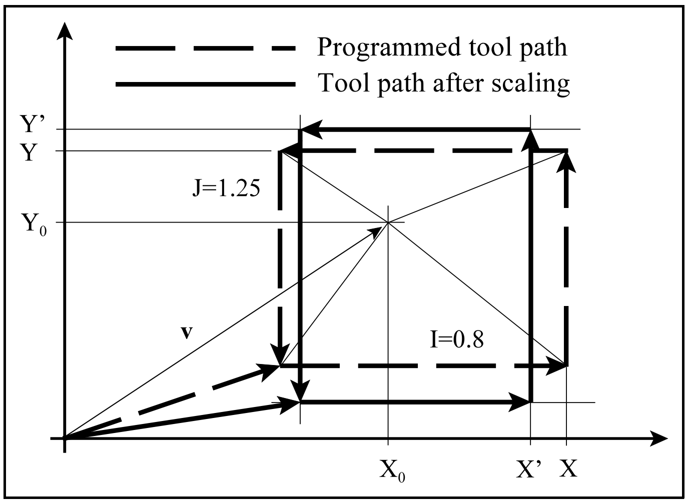


Fig. 16.2-3

The coordinates X, Y and Z of the center point of scaling can be specified as ***either absolute or incremental*** data by the use of G90, G91 or the operator I.

If value is not assigned to one of the axis addresses or each of them, the instantaneous axis position will be interpreted as the center of scaling.

At the address

I J K

the ***magnification rate for the axes I-X, J-Y and K-Z,*** respectively can be specified.

There will be reduction if I J K<1, and if I J K>1, magnification will occur.

The equation of the scaling of a shape in the space XYZ is the following:

$$X' = (X - X_0)I + X_0$$
$$Y' = (Y - Y_0)J + Y_0$$
$$Z' = (Z - Z_0)K + Z_0$$

where: X_0, Y_0, Z_0: the coordinates v of the center of scaling;
I J K: the magnification rate of scaling;
X, Y, Z: the coordinates of the programmed point;
X', Y', Z': the coordinates after scaling.

The command

G50

cancels the scaling.

☞ ***Notes:***

– The magnification rates of scaling (I, J and K) do not affect the value of radius compensation applied.

– The values of I, J and K can also be negative. In this case, not only scaling, but mirroring too will be executed by the control, along the given axis. If the values of I, J and K are -1, only mirroring will be executed, along the given axis.

– If the values of I, J and K are different, the circular interpolation (G1, G3) and the arc radius R will be modified as follows::

```
G17
G51 X0 Y0 Z0 I0.5 J0.75 K1.25
G1 X100 ,R10 (R=,R*(I+J)/2)
Y100
X10
G3 X0 Y90 R10 (R=R*I)
G1 Y0
G50
```

The control will multiply the arc radius by the average of I and J, and in the block G3 the circle radius by I.

16.3 Mirroring a Shape through One or More Straight Lines (G50.1, G51.1)

By the use of the command

G51.1 v

a programmed shape can be ***mirrored by the control through the coordinates selected*** in v.

In the coordinates

v

there can be specified ***that axis***, ***the straight line or lines parallel with which will be used for mirroring.***

The v can be the axis address X, Y and Z, and other axis addresses.

The position of the straight line has to be determined as follows: If the straight line the mirroring is to be done is parallel with the axis X, the position will have to be specified at the axis Y; if the straight line the mirroring is to be done is parallel with the axis Y, the position will have to be specified at the axis X,; and so on.

The entered ***coordinate data v*** will be interpreted by the control ***in the orthogonal coordinate system***, even if polar coordinate data specification is valid.

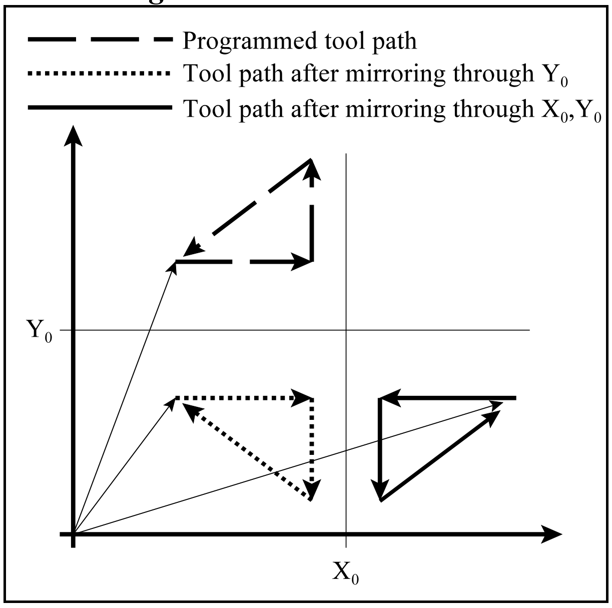

Programmed tool path
Tool path after mirroring through Y_0
Tool path after mirroring through X_0,Y_0
Y_0
X_0

Fig. 16.3-1

The coordinates vof the axes of mirroring can be specified as ***either absolute or incremental*** data by the use of G90, G91 or the operator I.

If no value is assigned to any of the axis addresses, the control will not execute mirroring in that one.

The equation of the scaling of a shape in the space XYZ is the following:

$$X' = -(X - X_0) + X_0$$

$$Y' = -(Y - Y_0) + Y_0$$

$$Z' = -(Z - Z_0) + Z_0$$

where: X_0, Y_0, Z_0: the coordinates v of the center point of mirroring;
X, Y, Z: the coordinates of the programmed point;
X', Y', Z': the coordinates after mirroring.

Mirroring through the axes of odd number will change the direction of motion around the shape. As a result of this, the circle directions (G2-G3), the direction of tool radius compensation (G41-G42) and the angle of rotation (G68R) also will change, and this change will automatically be taken into account by the control.

The command

G50.1 v

cancels mirroring through coordinate axis (axes) specified in v. In the coordinates, any data can be written, the effect will be only recording the fact of cancellation.

When activating or cancelling the mirroring, neither rotation (G68) nor scaling (G51) has to be in effect. Otherwise, the control sends the error message '2001 Unable to turn on or off mirroring under G51 or G68'.

Mirroring in the case of double tool holder

If, on the lathe, there are two tool changers, one of them is in the upper half space X+ and the another is in the lower half space X−, then the mirroring through the axis Z

G51.1 X0

can be used in the course of machining performed with the tools being in the lower tool changer.

If the machining is to be performed by the use of the tools of the tool changer being in the lower half space X−, the program can be written for the half space X+, and then the program part can be mirrored through the axis Z.

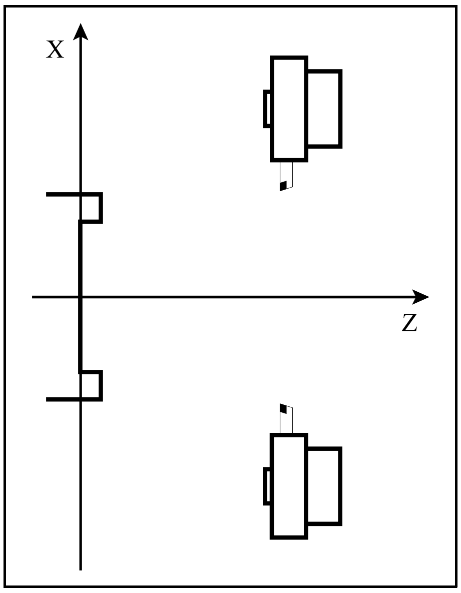


Fig. 16.3-2

An example:

Finishing the workpiece is to be executed by the use of the tool T2 being in the lower tool changer. The program has to be written in the way as if the tool was in the upper tool changer and then the path has to be mirrored through the axis Z.

```
G0 X140 Z50
T202 M3 (finishing tool, Q2)
G51.1 X0
N1 G0 G42 X20 Z10
Z0
X40
Z-20
G2 X60 Z-30 R10
G1 X80
Z-40
X100 Z-50
X140
N2 G40 G0 Z50
G50.1 X0
N3 X140
```

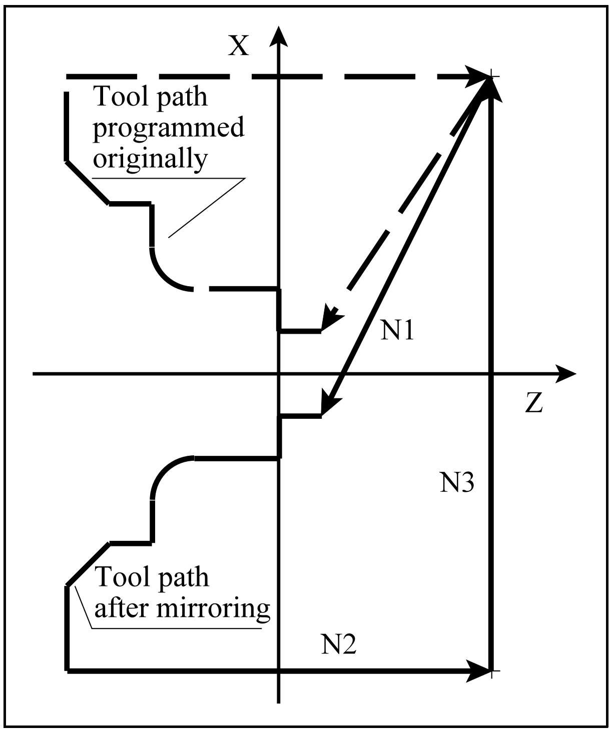


Fig. 16.3-3

16.4 Programming Rules for Specific Transformations

The sequence of the rotation G68 and the scaling G51 is optional.
However, attention should be payed if rotation is done first and scaling follows, because ***the command of rotation will be valid for the coordinates of the center point of scaling***. But, if scaling is done first and rotation follows, ***the command of scaling will be valid for the coordinates of the center point of rotation.***
The starting and cancelling commands of both operations have to be nested into one another; overlapping one another is not permitted:

Rotation–scaling

```
N1 G90 G17 G0 X0 Y0
N2 G68 X80 Y20 R75
N3 G51 X130 Y50 P0.5
N4 X180 Y20
N5 G1 Y80 F200
N6 X80
N7 Y20
N8 X180
N9 G50
N10 G69 G0 X0 Y0
```

Scaling–rotation

```
N1 G90 G17 G0 X0 Y0
N2 G51 X130 Y50 P0.5
N3 G68 X80 Y20 R75
N4 X180 Y20
N5 G1 Y80 F200
N6 X80
N7 Y20
N8 X180
N9 G69
N10 G50 G0 X0 Y0
```

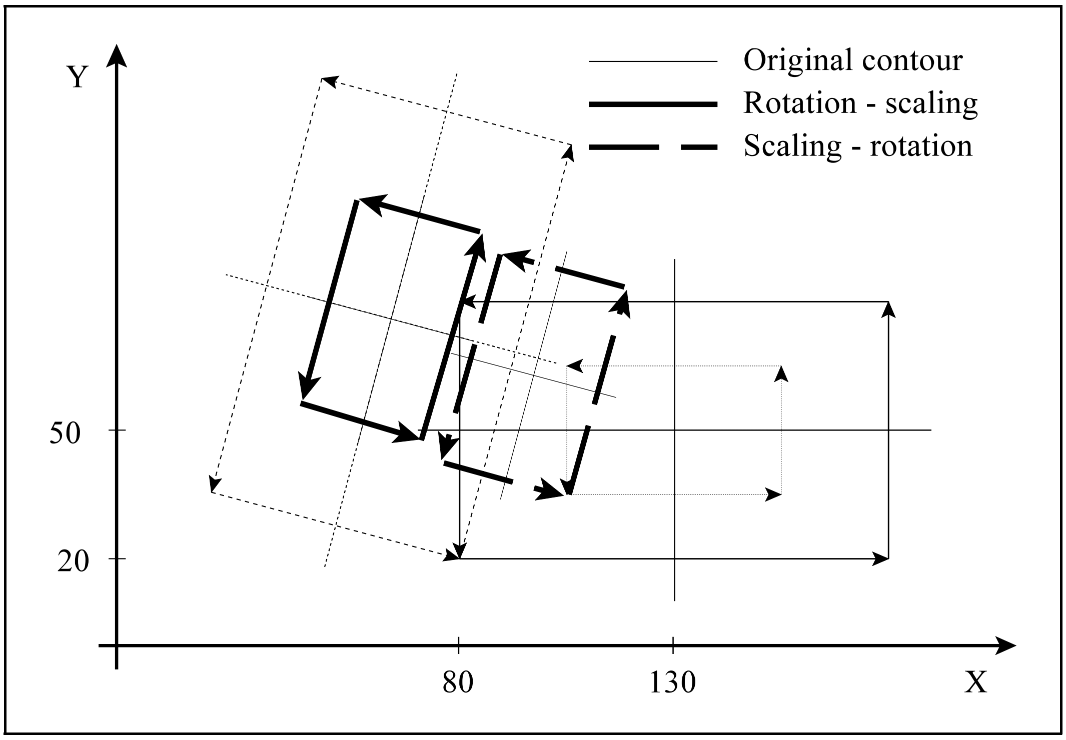


Fig. 16.4-1

It can be seen from the figure, that application sequence of several transformations cannot be disregarded.

Mirroring is quite a different matter. Starting up the mirroring is allowed in states G50 and G69 only, i.e. when there is neither scaling nor rotation states.

However, either scaling or rotation can be started up in the started up state of the mirroring.
It is valid for mirroring too, that overlapping of mirroring with either scaling or rotation is not allowed, so first the rotation and the scaling have to be cancelled in proper sequence, and the mirroring only after that.

```
G51.1 ...       (starting up of the mirroring)
G51 ...         (starting up of the scaling)
G68 ...         (starting up of the rotation)
...
G69 ...         (cancelling of the rotation)
G50 ...         (cancelling of the scaling)
G50.1 ...       (cancelling of the mirroring)
```

17 Automatic Geometric Calculations

17.1 Programming Chamfer and Corner Rounding

The control can insert ***chamfer*** or ***corner rounding*** automatically ***in the selected plane*** between two blocks containing linear interpolation (G01) or circular interpolation (G02, G03).

An isosceles chamfer, the side length of which is specified at the address

,C

(comma and C) is inserted by the control between the endpoint of the block containing the address ,C and the starting point of the succeeding block.

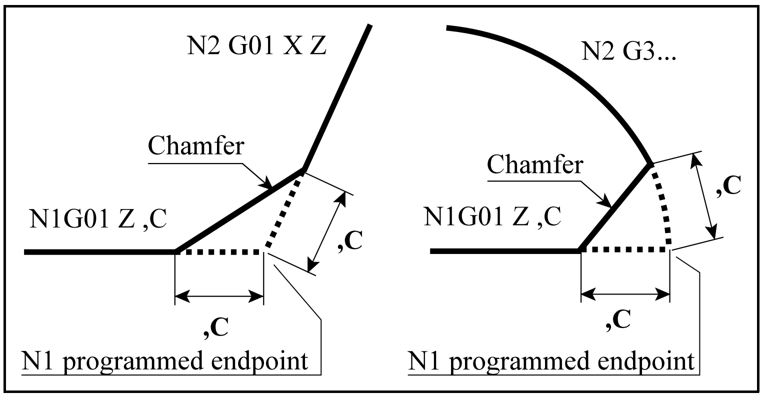


Fig. 17.1-1

For example:

```
      N1 G18 G1 G91 Z30 ,C10
      N2 X80 Z10
```

The value specified at the address ,C shows the distance, with which the chamfer starts and ends from the supposed intersection point of the two sequence blocks. Chamfer can also be inserted between circles or circle and straight line. In this case, the value of ,C is the length of the chord drawn from the intersection point.

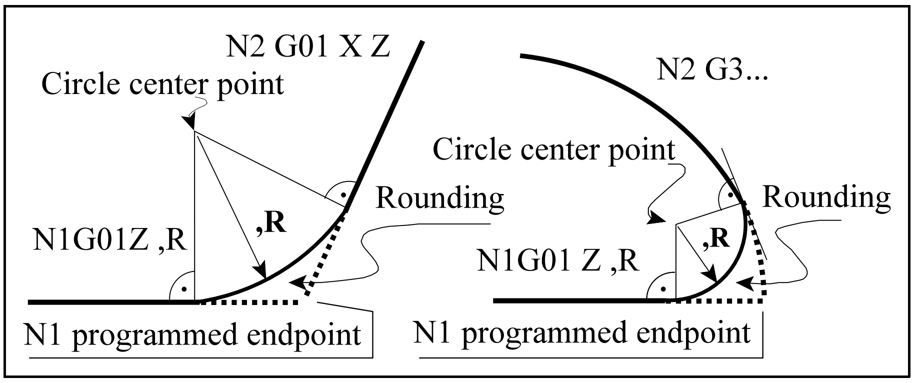


Fig. 17.1-2

A corner rounding, the radius of which is specified at the address

,R

(comma and R) is inserted by the control between the endpoint of the block containing the address ,R and the starting point of the succeeding block.

For example:

```
      N1 G18 G91 G01 Z30 ,R8
      N2 G03 X60 Z-30 R30
```

The control inserts the circle arc with the radius of ,R between the two blocks in such a way, that the circle osculates tangentially to the both path elements.

A command containing chamfer or corner rounding can also be written at the end of several succeeding blocks, as it is illustrated by the example below:

```
...
G18 G1 X80 ,C10
Z60 ,R22
G3 X160 Z20 R40 ,C10
G1 X220
...
```

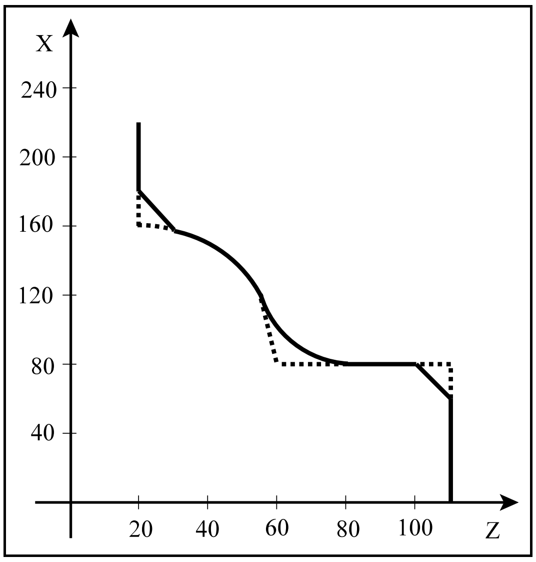


Fig. 17.1-3

☞ ***Notes***:

– Chamfer or corner rounding can only be programmed between elements being in the selected plane (G17, G18, G19), otherwise the control sends error message '2085 Illegal ,C or ,R'.

– If the side length of the chamfer or the radius of the corner rounding is so great that it cannot be suited to the programmed blocks, the control will send the error message '2083 ,C or ,R' is too high.

– If both ,C and ,R is programmed within one block, the control will send the error message '2082 Whether chamfer or rounding'.

– In single block mode, the control stops and gets into the stop state after execution of chamfering or corner rounding.

17.2 Specification of a Straight Line Using its Angle of Inclination

A straight line in the plane defined by the commands G17, G18, G19 ***can be specified by one of the coordinates of the selected plane and by its angle of inclination given at the address ,A***.

$$G17\begin{Bmatrix} G0 \\ \overline{G1} \end{Bmatrix}\begin{Bmatrix} X_p\,,A \\ \overline{Y_p\,,A} \end{Bmatrix}qF$$

$$G18\begin{Bmatrix} G0 \\ \overline{G1} \end{Bmatrix}\begin{Bmatrix} Z_p\,,A \\ \overline{X_p\,,A} \end{Bmatrix}qF$$

$$G19\begin{Bmatrix} G0 \\ \overline{G1} \end{Bmatrix}\begin{Bmatrix} Y_p\,,A \\ \overline{Z_p\,,A} \end{Bmatrix}qF$$

In the formulas above, X_p, Y_p and Z_p are the endpoint coordinates of the straight line along the axes X, Y and Z or along the axes parallel with them; q indicates one or more arbitrary axes being out of the selected plane.
Specification at the address ,A can also be used in addition to the codes G0 and G1.
The angle ,A is measured from the first axis of the selected plane, and the positive direction is opposite to the clockwise direction.
The value of ,A can be either positive or negative, and it can also be greater than 360°or smaller than −360°.

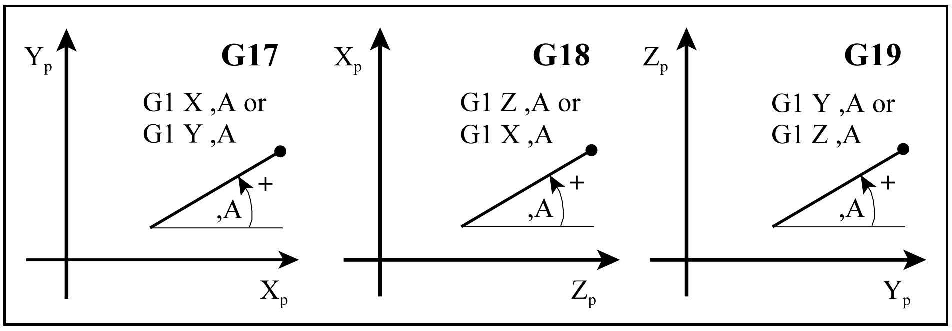


Fig. 17.2-1

For example:

```
(G18 G90) G1 X60 Z120 ...
Z70 ,A150
```
(this specification is equivalent to the specification X117.735 Z70)

```
X180 ,A135
```
(this specification is equivalent to the specification X180 Z38.868)

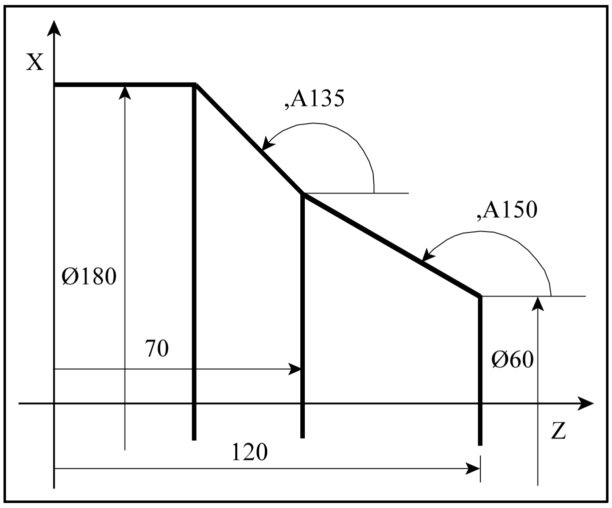


Fig. 17.2-2

☞ ***Notes:***

– Straight line with its angle of inclination and chamfer or corner rounding can also be specified in one block. For example:
```
Z100 ,A30 ,C5
X100 ,A120 ,R10
Z-100 ,A210
```
– Specification of the angle of inclination at the address ,A can be used in drilling cycles, too. In this case, it is taken into account by the control during execution of positioning in the selected plane in the way described above. For example, the block
```
G81 G91 X100 ,A30 R-2 Z-25
```
is equivalent to the following block:
```
G81 G91 X100 Y57.735 R-2 Z-25
```

17.3 Calculations of Intersection Point in the Plane

The calculations described here are executed by the control in the only case, ***when the calculation of tool radius compensation (G41 or G42 offset mode) is on.*** If tool radius compensation is not to be taken into account in the program, even this case it has to be switched on and the value of adequate tool radius compensation has to be deleted.

17.3.1 Linear-Linear Intersection

If there are two succeeding blocks executing linear interpolation, and the ***second straight line*** is defined ***by specification the endpoint of the line on both axes in the selected plain and the angle of inclination of the line too***, the control will calculate the intersection point of the straight line assigned in the first block and the straight line specified in the second block.
Hereafter, the straight line specified in the second block in such a way is called ***overdetermined straight line***.

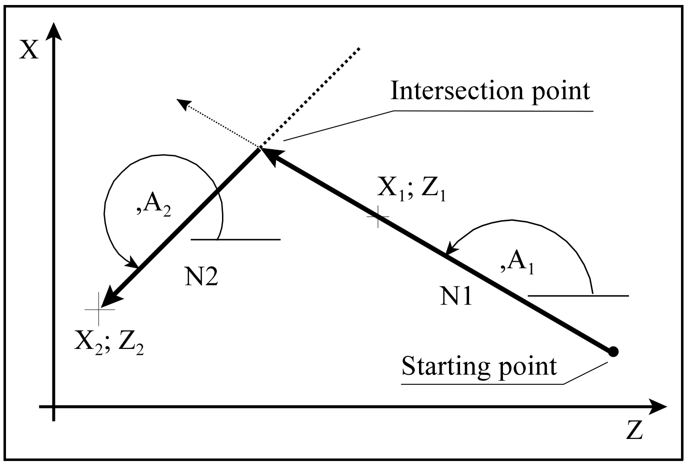


Fig. 17.3.1-1

The endpoint of the first block and the starting point of the second block will be the calculated intersection point.

G17 G41 (G42)	G18 G41 (G42)	G19 G41 (G42)
N1 G1 ,A_1 or	N1 G1 ,A_1 or	N1 G1 ,A_1 or
X_1 Y_1	X_1 Z_1	Y_1 Z_1
N2 G1G90 $\mathbf{X_2}$ $\mathbf{Y_2}$ **,$\mathbf{A_2}$**	N2 G1G90 $\mathbf{X_2}$ $\mathbf{Z_2}$ **,$\mathbf{A_2}$**	N2 G1G90 $\mathbf{Y_2}$ $\mathbf{Z_2}$ **,$\mathbf{A_2}$**

The intersection point is always calculated in the plane selected by the G17, G18 and G19.
The ***first block*** (N1) is specified either ***by its angle of inclination (,A) only***, and in this case the control draws a straight line from the starting point up to the intersection point at the appropriate angle of inclination; ***or***, ***an arbitrary point of the straight line*** (X_1, Y_1; X_1, Z_1; or Y_1, Z_1) different from the starting point ***is specified***, and in this case the control calculates the intersection point using the straight line passing through these two points.
The ***coordinates*** specified ***in the second block*** (N2) are always interpreted by the control as ***absolute*** data (G90).

For example:

```
    (G18) G90 G41 ...
    G0 X20 Z90
    N10 G1 ,A150
    N20 X40 Z10 ,A225
    G0 Z0
    ...
```

The block N10 can also be given using the coordinates of a point of the straight line:

```
    (G18) G90 G41 ...
    G0 X20 Z90
    N10 G1 X66.188 Z50
    N20 X40 Z10 ,A225
    G0 X0 Y20
    ...
```

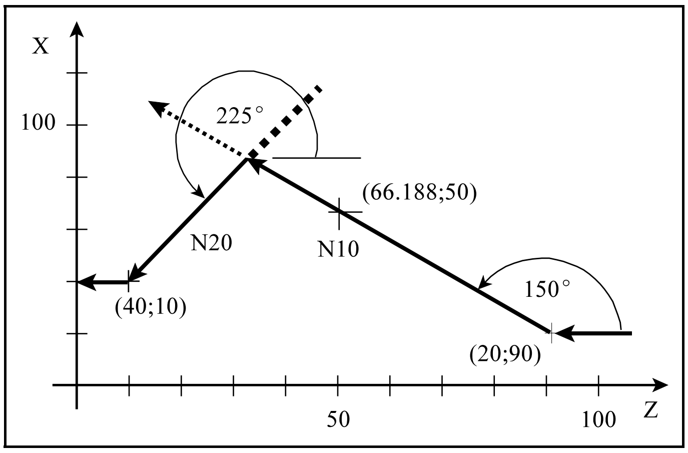


Fig. 17.3.1-2

It can be noted, that in this case the coordinates X and Z (X66.18 Z50) given in the block N10 are not considered by the control to be endpoint, but as a transition point only between the starting point of the straight line and the point given.

Calculation of intersection point can also be combined with specification of chamfer or corner rounding. For example:

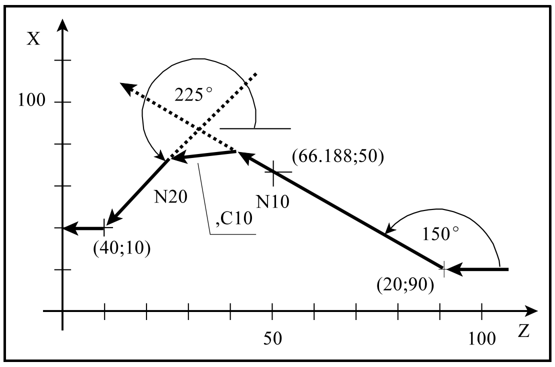


Fig. 17.3.1-3

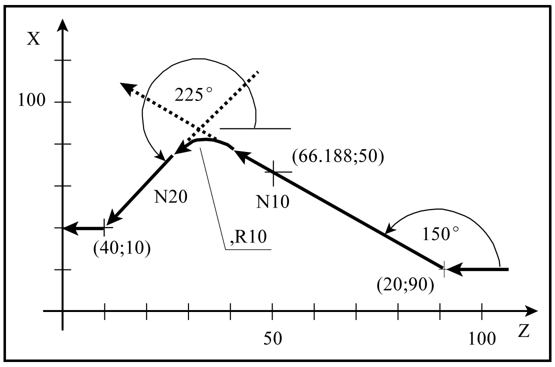


Fig. 17.3.1-4

```
(G18) G90 G41 ...
G0 X20 Z90
N10 G1 X66.188 Z50 ,C10
N20 X40 Z10 ,A225
G0 X0 Y20
...
```

```
(G18) G90 G41 ...
G0 X20 Z90
N10 G1 X66.188 Z50 ,R10
N20 X40 Z10 ,A225
G0 X0 Y20
...
```

In the examples above, the control measures the length of chamfer from the calculated intersection point and fits the corner rounding to the calculated intersection point, respectively.

17.3.2 Linear-Circular Intersection

If a circular block is given after a linear block in a way that ***the coordinates of the end point and the center point as well as the radius of the circle are specified, i.e. the circle is overdetermined***, the control will calculate intersection point between the straight line and the circle. ***The calculated intersection point is the end point of the first block and the start point of the second one.***

G17 G41 (G42)	G18 G41 (G42)	G19 G41 (G42)
N1 G1 ,A or	N1 G_1,A or	N1 G1 ,A or
X_1 Y_1	X_1 Z_1	Y_1 Z_1
N2 G2 (G3) G90 X_2 Y_2 I J R Q	N2 G2 (G3) G90 X_2 Z_2 I K R Q	N2 G2 (G3) G90 Y_2 Z_2 J K R Q

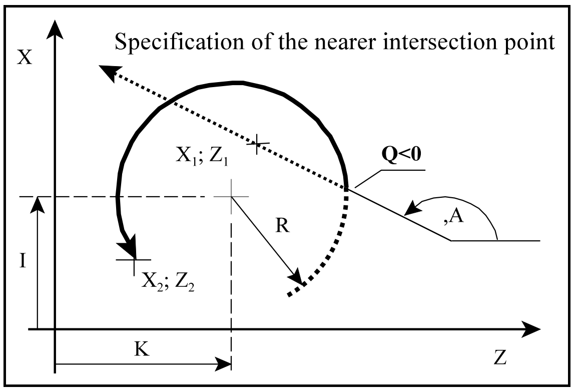


Fig. 17.3.2-1

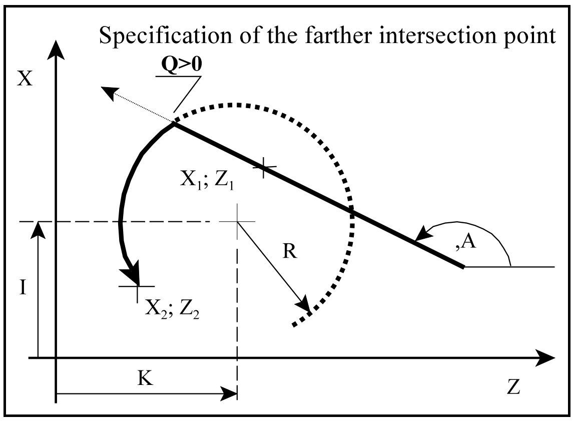


Fig. 17.3.2-2

The intersection point is always calculated by the control in the plane selected by G17, G18 and G19.

The ***first block*** (N1) is specified either ***by its angle of inclination (,A) only***, and in this case the control draws a straight line from the starting point up to the intersection point at the appropriate angle of inclination; ***or, an arbitrary point of the straight line*** (X_1, Y_1; X_1, Z_1; or Y_1, Z_1) different from the starting point ***is specified***, and in this case the control calculates the intersection point using the straight line passing through these two points.

The ***coordinates*** specified ***in the second block*** (N2), thus ***the coordinates I, J and K*** defining ***the center point of the circle*** are also always interpreted by the control as ***absolute*** data (G90).

It can be specified ***at the address Q***, which one between the two resulting intersection points is to be calculated by the control. ***If the value of the address is smaller than zero (Q<0), the control will calculate the nearer intersection point in the direction of the straight line, however, if the value of the address is greater than zero (Q>0), the farther one will be calculated***. The direction of movement along the straight line is determined by the angle of inclination.

Let us see the following example:

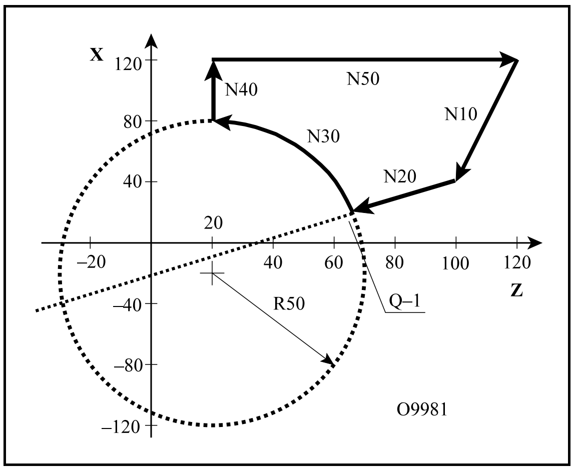


Fig. 17.3.2-3

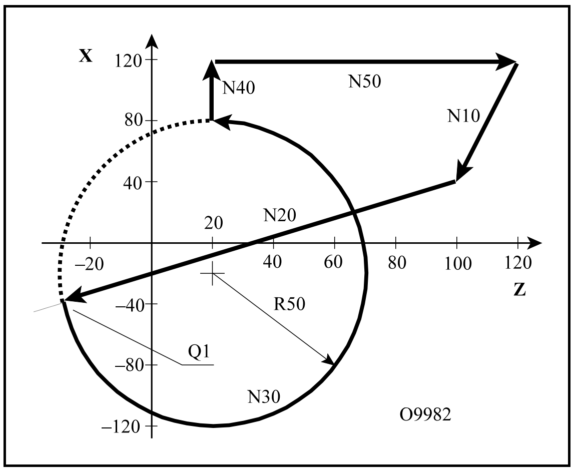


Fig. 17.3.2-4

```
O9981
N10 (G18) G42 G0 X40 Z100
N20 G1 X-40 Z-30
N30 G3 X80 Z20 I-10K20 R50 Q-1
N40 G40 G0 X120
N50 Z120
```

```
O9982
N10 (G18) G42 G0 X40 Z100
N20 G1 X-40 Z-30
N30 G3 X80 Z20 I-10 K20 R50 Q1
N40 G40 G0 X120
N50 Z120
```

The circular block N30 G3 is an overdetermined one since the coordinates of the center point of the circle (I–10 K20 in absolute value) as well as the radius of the circle (R50) are specified; the intersection point of the straight line specified in the block N20 and the circle specified in the block N30 will be calculated by the control.
In the program O9981, the control calculates the nearer intersection point in the direction of the straight line, because Q–1 is programmed in the circle block N30.
However, in the program O9982, the control calculates the farther intersection point in the direction of the straight line, because Q1 is programmed in the circle block N30.
The linear-circle intersection calculation can also be combined with specification of chamfer or corner rounding. For example:

```
    N10 (G18) G42 G0 X40 Z100 S200 M3
    N20 G1 X-40 Z-30 ,R15
    N30 G3 X80 Z20 I-10 K20 R50 Q-1
    N40 G40 G0 X120
    N50 Z120
```

The control calculates the intersection point of the blocks N20 and N30, and then, due to the ,R15 given in the blockN20, it fits a corner rounds with radius of 15 mm to the intersection point.

17.3.3 Circular-Linear Intersection

If a linear block is given after a circular block in a way that ***the straight line is overdetermined***, i.e. ***the coordinate of the endpoint as well as the angle of inclination of the straight line are specified***, the control will calculate intersection point between the circle and the straight line. ***The calculated intersection point is the end point of the first block and the start point of the second one.***

G17 G41 (G42)	G18 G41 (G42)	G19 G41 (G42)
N1 G2 (G3) X_1 Y_1 I J	N1 G2 (G3) X_1 Z_1 I K	N1 G2 (G3) Y_1 Z_1 J K
or R	or R	or R
N2 G1 G90 X_2 Y_2 ,A Q	N2 G1 G90 X_2 Z_2 ,A Q	N2 G1 G90 Y_2 Z_2 ,A Q

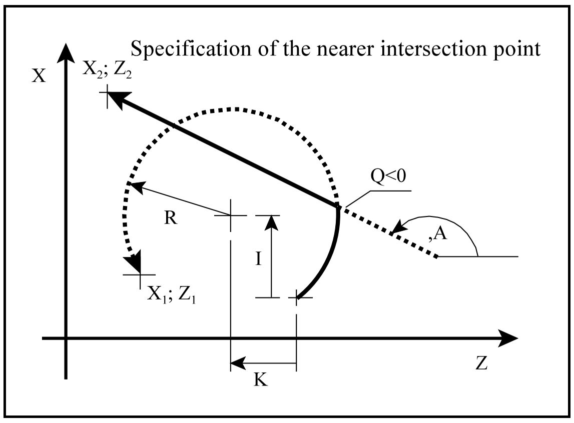


Fig. 17.3.3-1

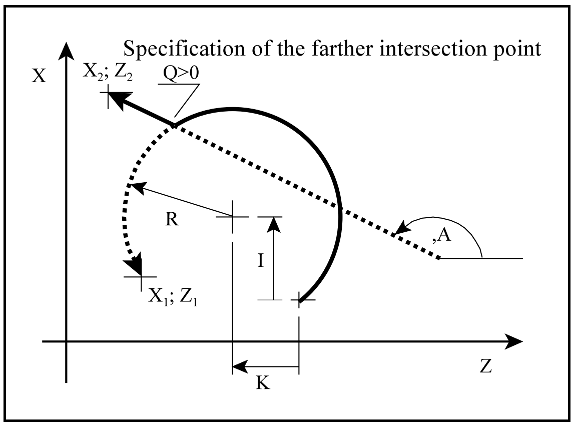


Fig. 17.3.3-2

The intersection point is always calculated by the control in the plane selected by G17, G18 and G19.

The first block (N1), i.e. the circle is specified either by ***one arbitrary pont of it*** (X_1, Y_1; X_1, Z_1; or Y_1, Z_1) as well as ***the coordinates of its center point*** (I J; I K; or J K), ***or*** instead of the coordinates of the center point, ***the radius (R) of the circle also can be specified***.

In the second block (N2), ***the straight line is overdetermined***, i.e. ***its endpoint coordinates (X_2 Y_2; X_2 Z_2; orY_2 Z_2) and its angle of inclination (,A) are specified***. The ***endpoint coordinates*** of the straight line are always interpreted by the control as ***absolute*** data (G90). It is always ***the angle of inclination of the straight vector pointing from the resulting intersection point to the given endpoint*** has to be specified at the address ,A; otherwise motions opposite to the intended ones will occur.

It can be specified ***at the address Q***, which one between the two resulting intersection points is to be calculated by the control. ***If the value of the address is smaller than zero (Q<0, for example Q–1), the control will calculate the nearer intersection point in the direction of the straight line; however, if the value of the address is greater than zero (Q>0, for example Q1), the farther one will be calculated***. ***The direction of movement along the straight line is determined by the angle of inclination.***

Let us see the following example:

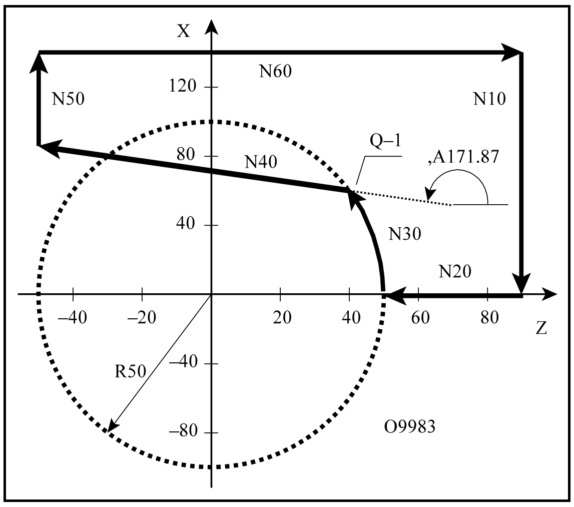


Fig. 17.3.3-3

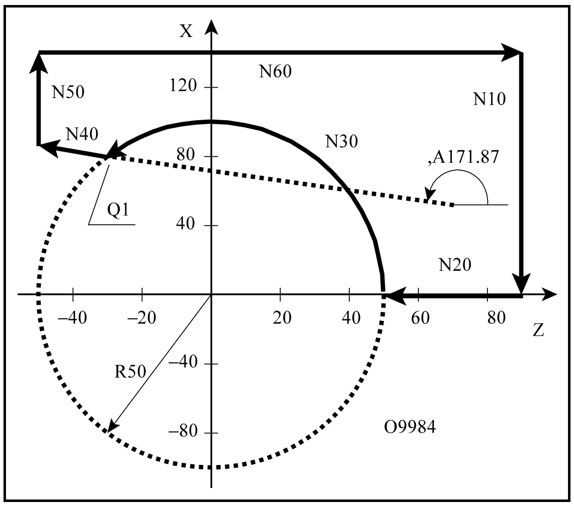


Fig. 17.3.3-4

```
O9983
N10 (G18) G0 X0 Z90
N20 G42 G1 Z50
N30 G3 X0 Z-50 R50
N40 G1 X85.714 Z-50 ,A171.87
Q-1
N50 G40 G0 X140
N60 Z90
```

```
O9984
N10 (G18) G0 X0 Z90 M3 S200
N20 G42 G1 Z50
N30 G3 X0 Z-50 R50
N40 G1 X85.714 Z-50 ,A171.87
Q1
N50 G40 G0 X140
N60 Z90
```

The linear block N40 is an overdetermined one, since the endpoint coordinates of the straight line (X85.714 Z–50) as well as its angle of inclination (,A171.87) are specified. For this reason, the coordinates X0 Z–50 of the circle programmed in the previous block N30 are not considered by the control to be endpoint values but as a point only through which the circle passes, and the endpoint will be the calculated intersection point.
In the program O9983, the nearer intersection point (Q–1) in the direction of motion is given, while in the program O9984 the farther one (Q1) is given.
The circle-linear intersection calculation can also be combined with specification of chamfer or corner rounding. For example:

```
    O9983
    N10 (G18) G0 X0 Z90 M3 S200
    N20 G42 G1 Z50
    N30 G3 X0 Z-50 R50 ,R15
    N40 G1 X85.714 Z-50 ,A171.87 Q-1
    N50 G40 G0 X140
    N60 Z90
```

In this example, a corner rounding with radius of 15 mm (,R15) is specified in the block N30. The control calculates the intersection point of the blocks N30 and N40, and fits the programmed corner rounding to the resulting contour.

17.3.4 Circular-Circular Intersection

If two succeeding circular blocks are given ***by specification of the endpoint and the center point coordinates as well as the radius of the second circle***, i.e. ***the second circle is overdetermined***, the control will calculate intersection point between the two circles. ***The calculated intersection point is the end point of the first block and the start point of the second one.***

G17 G41 (G42)	G18 G41 (G42)	G19 G41 (G42)
N1 G2 (G3) X_1 Y_1 I_1 J_1	N1 G2 (G3) X_1 Z_1 I_1 K_1	N1 G2 (G3) Y_1 Z_1 J_1 K_1
or X_1 Y_1 R_1	or X_1 Z_1 R_1	or Y_1 Z_1 R_1
N2 G2 (G3) G90 X_2 Y_2 I_2 J_2 R_2 Q	N2 G2 (G3) G90 X_2 Z_2 I_2 K_2 R_2 Q	N2 G2 (G3) G90 Y_2 Z_2 J_2 K_2 R_2 Q

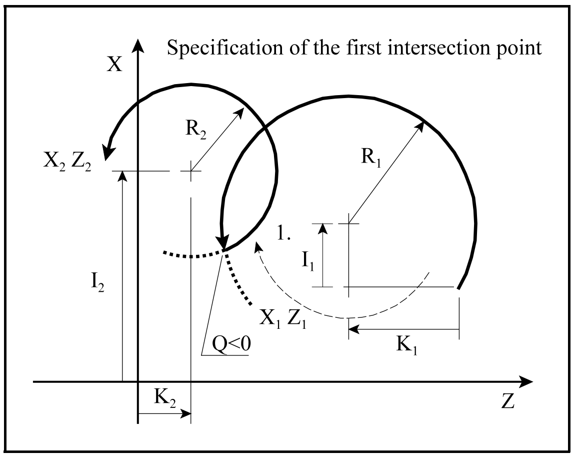


Fig. 17.3.4-1

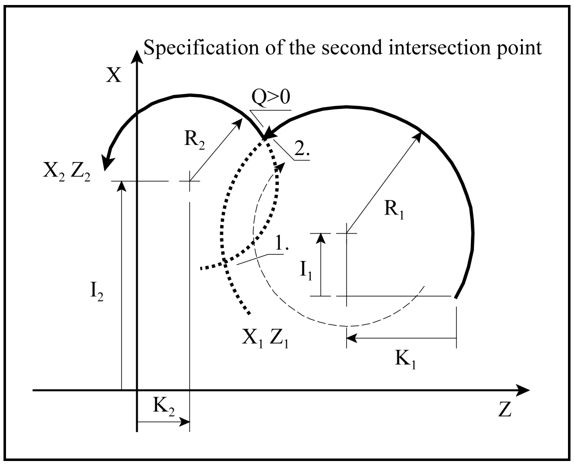


Fig. 17.3.4-2

The intersection point is always calculated by the control in the plane selected by G17, G18 and G19.
The first block (N1) is specified either ***by the coordinates of the center point of the circle*** (I_1 J_1; I_1 K_1; J_1 K_1) or ***by its radius*** (R_1). In this block, the coordinates are interpreted as the relative distance measured from the starting point, like specification of a circle.
The ***coordinates*** specified ***in the second block*** (N2), thus ***the coordinates I, J and K*** defining ***the center point of the circle*** are also always interpreted by the control as ***absolute*** data (G90).
It can be specified ***at the address Q***, which one between the two resulting intersection points is to be calculated by the control. ***If the value of the address is smaller than zero (Q<0, for example Q–1), the control will calculate the first intersection point; however, if the value of the address is greater than zero (Q>0, for example Q1), the second one will be calculated***.
The first intersection point is that one being passed through at first, going clockwise (independently of the programmed direction G2, G3).

Let us see the following example:

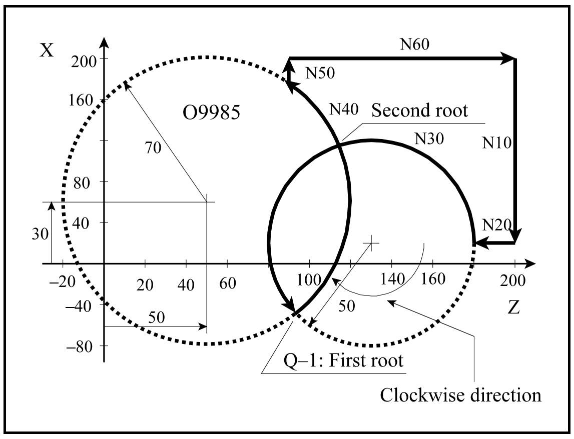


Fig. 17.3.4-3

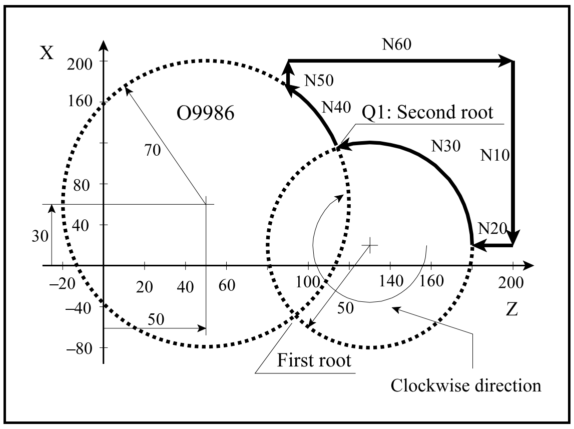


Fig. 17.3.4-4

```
O9985
N10 (G18) G0 X20 Z200
N20 G42 G1 Z180
N30 G3 X-80 Z130 R-50
N40 X174.892 Z90 I30 K50 R70
Q-1
N50 G40 G0 X200
N60 Z200
```

```
O9986
N10 (G18) G0 X20 Z200
N20 G42 G1 Z180
N30 G3 X-80 Z130 R-50
N40 X174.892 Z90 I30 K50 R70
Q1
N50 G40 G0 X200
N60 Z200
```

The circular block N40 is an overdetermined block since the coordinates of the center point (I30 K50 as absolute value, and I as radius) as well as the radius (R70) is specified. For this reason, the coordinates X-80 Z130 of the circle programmed in the previous block N30 are not considered by the control to be endpoint values but as a point only through which the circle passes, and the endpoint will be the calculated intersection point.
In the program O9985, the nearer intersection point (Q–1) in the clockwise direction, while in the program O9986 the farther one (Q1) is given.
The circle-circle intersection calculation can also be combined with specification of chamfer or corner rounding. For example:

```
O9986
N10 (G18) G0 X20 Z200 M3 S200
N20 G42 G1 Z180
N30 G3 X-80 Z130 R-50 ,R20
N40 X174.892 Z90 I30 K50 R70 Q1
N50 G40 G0 X200
N60 Z200
```

In this example, a corner rounding with radius of 20 mm (,R20) is specified in the block N30. The control calculates the intersection point of the blocks N30 and N40, and fits the programmed corner rounding to the resulting contour.

17.3.5 Chaining the Intersection Calculations

Blocks of intersection point calculation can be chained, i.e. ***several succeeding blocks can be selected for intersection point calculation***. The control ***calculates intersection point until it finds overdetermined straight lines or circles***.
Let us see the following example:

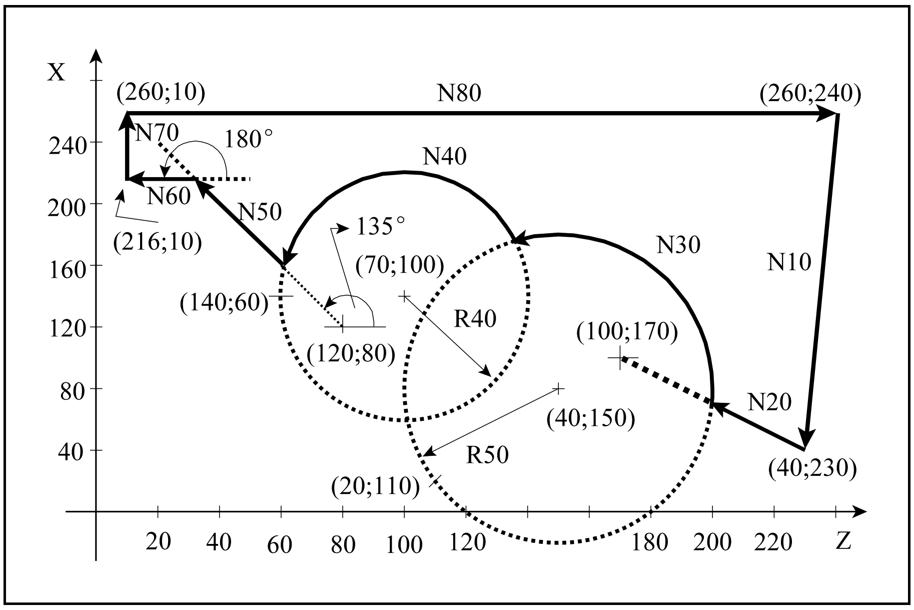


Fig. 17.3.5-1

```
N10 (G18) G0 G42 X40 Z230 F300
N20 G1 X100 Z170
N30 G3 X20 Z110 I40 K150 R50 Q-1
N40 X140 Z60 I70 K100 R40 Q1
N50 G1 X120 Z80 ,A135 Q1
N60 X216 Z10 ,A180
N70 G40 G0 X260
N80 Z240
```

The blocks N30, N40, N50 and N60 are overdetermined ones.
The straight line N20 is not guided by the control up to its programmed endpoint (X100 Z170) because the block N30 is overdetermined, i.e. all the addresses I K and R are filled and the intersection point to be found is specified at the address Q.
Neither the circle block N30 is guided up to the programmed endpoint (X20 Z110) because the circular N40 is overdetermined, too.
The last overdetermined block in the program is the straight line N60. Since the succeeding linear block is not overdetermined, thus the coordinates X216 Z10 programmed in the block N60 are considered by the control not to be a transit point of the straight line, but as the endpoint coordinates of the block N60.
In general, those coordinate points of the overdetermined linear and circular blocks which are in the plane selected will only be considered by the control to be an endpoint coordinate if they are not followed by an overdetermined block.

18 Canned Cycles for Turning

18.1 Single Cycles

The single cycles are the longitudinal turning cycle G77, the simple thread turning cycle G78 and the face turning cycle G79.

18.1.1 Longitudinal Turning Cycle (G77)

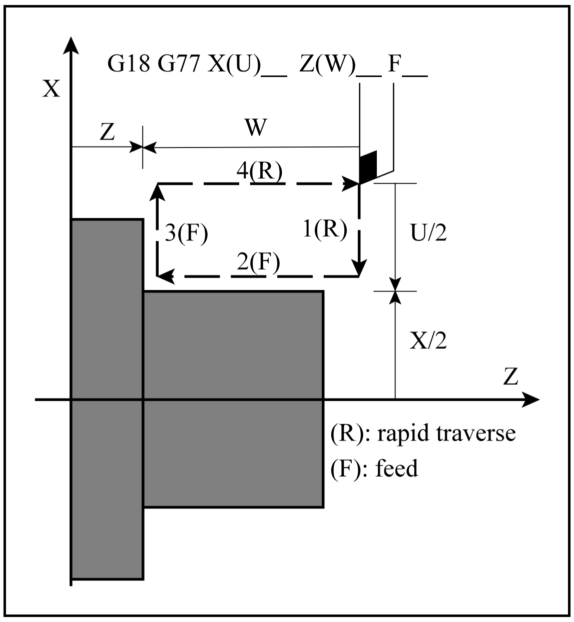


Fig. 18.1.1-1

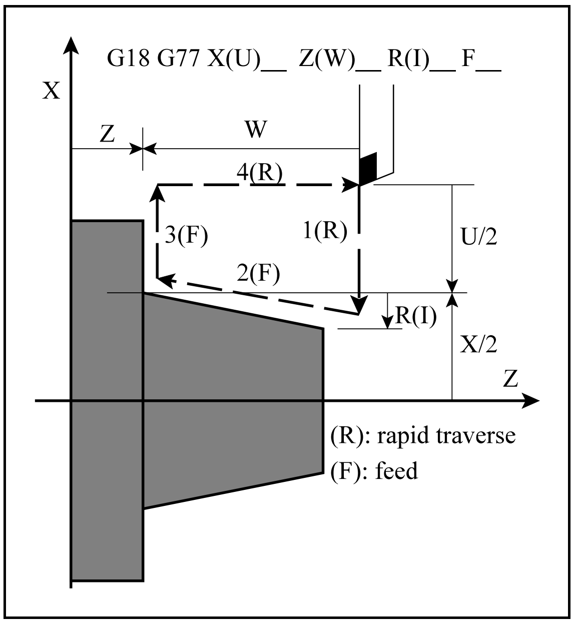


Fig. 18.1.1-2

The straight turning cycle can be defined in the following way:

G18 (plane Z-X)
G77 X_p(U)__ Z_p(W)__ F__

G19 (plane Y-Z)
G77 Y_p(V)__ Z_p(W)__ F__

G17 (plane X-Y)
G77 X_p(U)__ Y_p(V)__ F__

The tool always executes infeed motion along the axis No. 2 of the selected plane and cutting motion along the axis No. 1 of the selected plane.
Incremental data specification can be done using operator I, by programming G91, and by specification of the addresses U, V and W, too.
In the case of incremental data specification, the sign of the data determines the direction of the paths Nos. 1 and 2. In the figure above that shows longitudinal turning in the plane G18, the sign of both U and W is negative.

The tool moves along the paths Nos. 2 and 3 at a feed rate programmed at the address F in the block or being modal one; and along the paths Nos. 1 and 4 at rapid traverse rate.

The taper turning cycle can be defined in the following way:

G18 (plane Z-X)
  G77 X_p(U)__ Z_p(W)__ R(I)__ F__

G19 (plane Y-Z)
  G77 Y_p(V)__ Z_p(W)__ R(K)__ F__

G17 (plane X-Y)
  G77 X_p(U)__ Y_p(V)__ R(J)__ F__

The amount of taper can be specified either at the address R or at the addresses I, J and K depending on plane selected. In both cases, the data interpretation is the same. The data specified at the address R (I, J, K) is always incremental; it is to be interpreted from the position specified at the address of the axis No. 2 of the plane and given in radius. The direction of the taper is determined by the sign of the address R (I, J, K).
Interpretation of the other addresses is of the same as it was in the case of the straight turning cycle.
The code G77 and the data programmed in the block G77 are modal.
In the block by block mode, the tool stops at the end of all the four (1, 2, 3 and 4) actions.

In the case of incremental programming, the signs of the addresses U, W and R(I) affect the motion direction in the state G18 as follows:

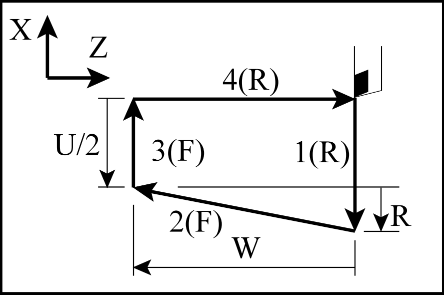
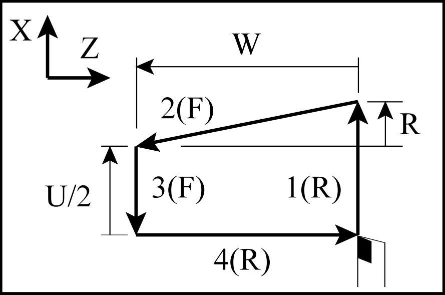
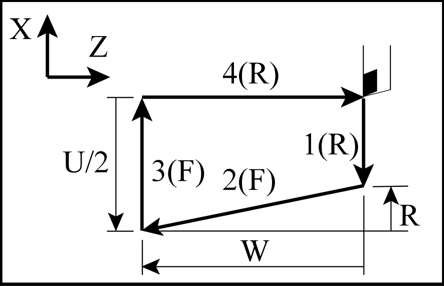
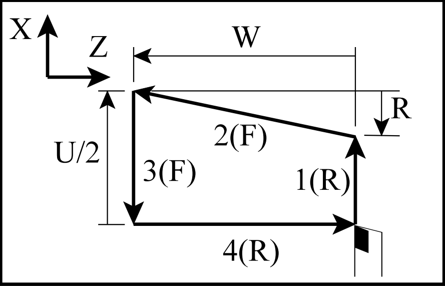
1. U<0, W<0, R<0	2. U>0, W<0, R>0
	
3. U<0, W<0, R>0, és $\lvert R\rvert \le \lvert U\rvert/2$	3. U>0, W<0, R<0, és $\lvert R\rvert \le \lvert U\rvert/2$
	

Fig. 18.1.1-3

18.1.2 Simple Thread Turning Cycle (G78)

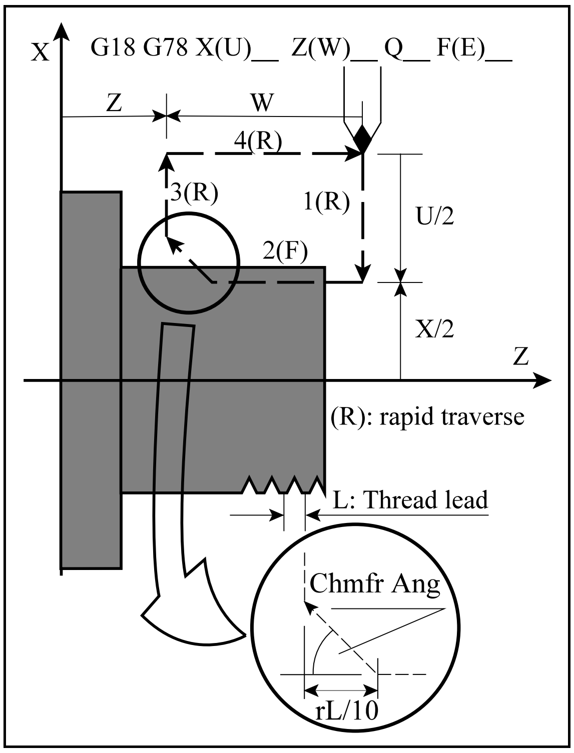


Fig. 18.1.2-1

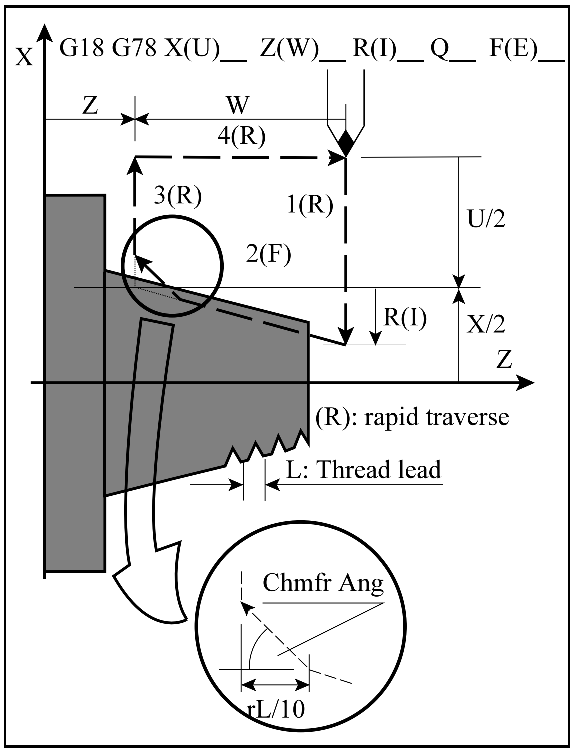


Fig. 18.1.2-2

The straight thread turning cycle can be defined in the following way:

G18 (plane Z-X)
G78 X_p(U)__ Z_p(W)__ Q__ F(E)__

G19 (plane Y-Z)
G78 Y_p(V)__ Z_p(W)__ Q__ F(E)__

G17 (plane X-Y)
G78 X_p(U)__ Y_p(V)__ Q__ F(E)__

The tool always executes infeed motion along the axis No. 2 of the selected plane and thread cutting motion along the axis No. 1 of the selected plane.
Incremental data specification can be done using operator I, by programming G91, and by specification of the addresses U, V and W, too.
In the case of incremental data specification, the sign of the data determines the direction of the paths Nos. 1 and 2. In the figure above that shows thread turning in the plane G18, the sign of both U and W is negative.
In the block, the tread lead is programmed at the address F or the number of threads per inch is programmed at the address E, and the angle of the thread start from the zero pulse of the encoder given in degree (°) is programmed at the address Q, as it is written in the block G33.
The motions 1, 3 and 4 are executed at a rapid traverse rate.
At the end of the path a chamfer with an angle set at the parameter N1607 Chmfr Ang is shaped.

The length of the chamfered section is determined by the parameter N1606 ThrdChmfr marked by r in the figure. The length of the section:

r·L/10

where: r: the value of the parameterN1606 ThrdChmfr;
L: the programmed thread lead.

If the value of the parameter is, for example 4, and the programmed thread lead is F2, the length of the chamfer section will be:

2*(4/10)=0.8 mm

The taper thread turning cycle can be defined in the following way:

G18 (plane Z-X)
G78 X_p(U)__ Z_p(W)__ R(I)__ Q__ F(E)__

G19 (plane Y-Z)
G78 Y_p(V)__ Z_p(W)__ R(K)__ Q__ F(E)__

G17 (plane X-Y)
G78 X_p(U)__ Y_p(V)__ R(J)__ Q__ F(E)__

The amount of taper can be specified either at the address R or at the addresses I, J and K depending on plane selected. In both cases, the data interpretation is the same. The data specified at the address ***R (I, J, K)*** is ***always incremental***; it is to be interpreted from the position specified at the address of the axis No. 2 of the plane and ***given in radius***. The direction of the taper is determined by the sign of the address R (I, J, K).
Interpretation of the other addresses is of the same as it was in the case of the straight thread turning cycle.
The angle of the chamfer is 45° in this case too, and the length of the chamfer r is measured along the straight line parallel with the axis.
The code G78 and the data programmed in the block G78 are modal.
In the block by block mode, the tool stops at the end of all the four (1, 2, 3 and 4) actions.

The effect produced by pushing the button STOP in the action 2 of the cycle

It is possible to stop the actions 1, 3 and 4 of the cycle any time using the button STOP, and the slides stop as they do in the case of the normal interpolation G0.
Pushing the button STOP is effective in the thread turning section too, however, in this case the control cuts such a chamfer at first it made at the end of the action 2, then it retracts at a rapid traverse rate along the axis X, after that it moves along the axis Z to the starting point. The button STOP is not effective already along the escape path.

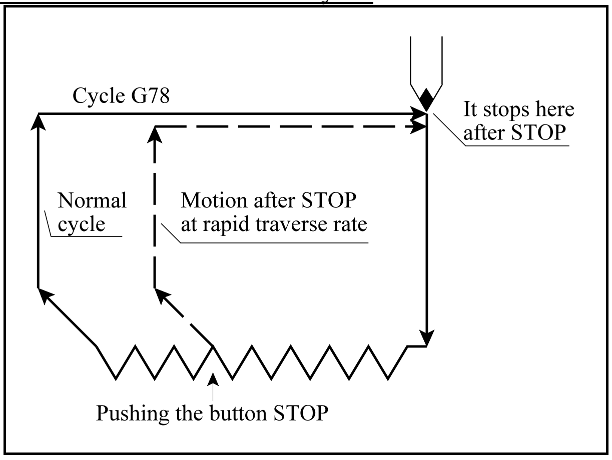


Fig. 18.1.2-3

18.1.3 Face Turning Cycle (G79)

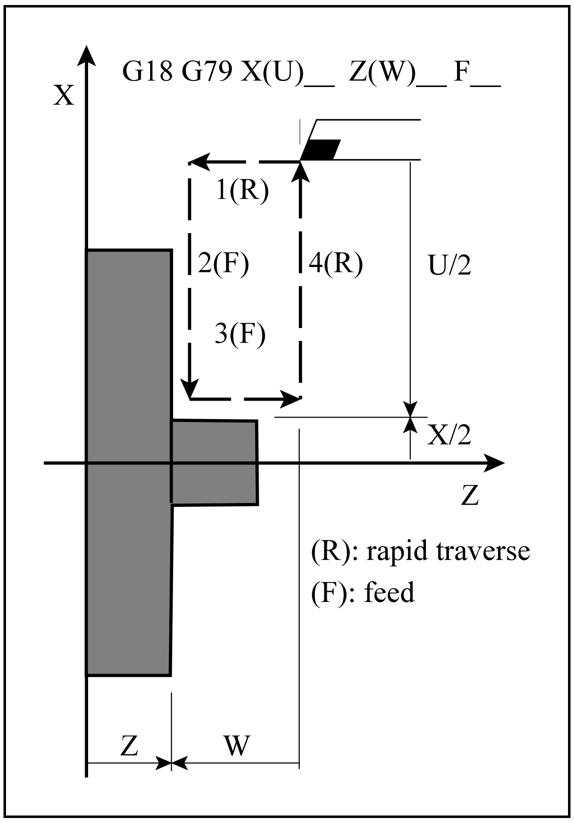


Fig. 18.1.3-1

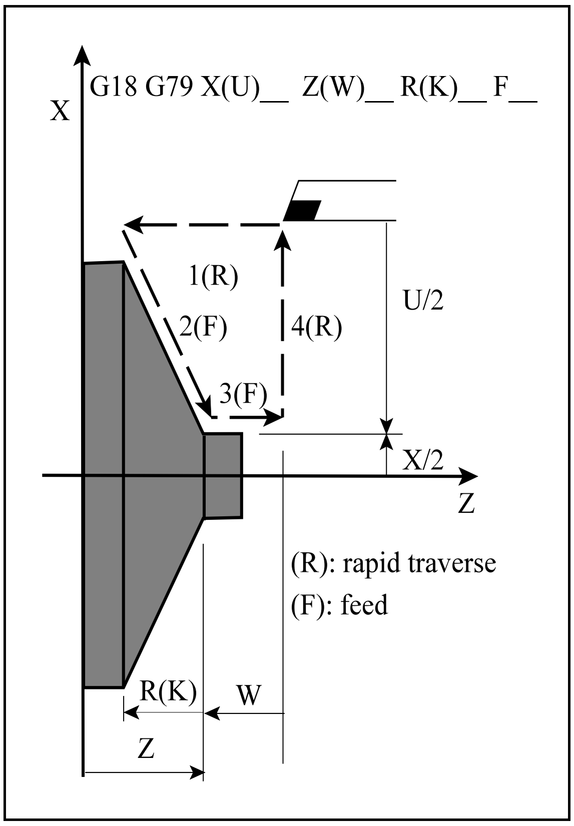


Fig. 18.1.3-2

The straight face turning cycle can be defined in the following way

G18 (plane Z-X)
G79 X_p(U)__ Z_p(W)__ F__

G19 (plane Y-Z)
G79 Y_p(V)__ Z_p(W)__ F__

G17 (plane X-Y)
G79 X_p(U)__ Y_p(V)__ F__

The tool always executes infeed motion along the axis No. 1 of the selected plane and cutting motion along the axis No. 2 of the selected plane.
Incremental data specification can be done using operator I, by programming G91, and by specification of the addresses U, V and W, too.
In the case of incremental data specification, the sign of the data determines the direction of the paths Nos. 1 and 2. In the figure above that shows face turning in the plane G18, the sign of both U and W is negative.
The tool moves along the paths Nos. 2 and 3 at a feed rate programmed at the address F in the block or being modal one; and along the paths Nos. 1 and 4 at rapid traverse rate.

The taper face turning cycle can be defined in the following way:

> **G18** (plane Z-X)
>> **G79** X_p(U)__ Z_p(W)__ R(K)__ F__
>
> G19 (plane Y-Z)
>> **G79** Y_p(V)__ Z_p(W)__ R(J)__ F__
>
> G17 (plane X-Y)
>> **G79** X_p(U)__ Y_p(V)__ R(I)__ F__

The amount of taper can be specified either at the address R or at the addresses I, J and K depending on plane selected. In both cases, the data interpretation is the same. The data specified at the address ***R (I, J, K)*** is ***always incremental***; it is to be interpreted from the position specified at the address of the axis No. 1 of the plane and ***given in radius***. The direction of the taper is determined by the sign of the address R (I, J, K).
Interpretation of the other addresses is of the same as it was in the case of the straight face turning cycle.
The code G79 and the data programmed in the block G79 are modal.
In the block by block mode, the tool stops at the end of all the four (1, 2, 3 and 4) actions.
In the case of incremental programming, the signs of the addresses U, W and R(K) affect the motion direction in the state G18 as follows:

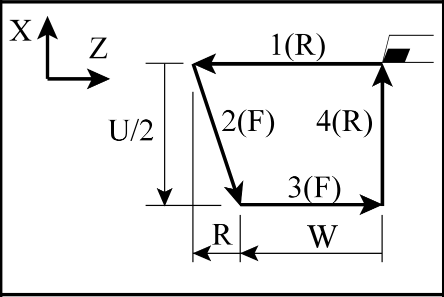
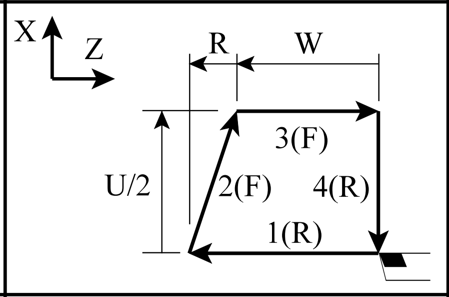
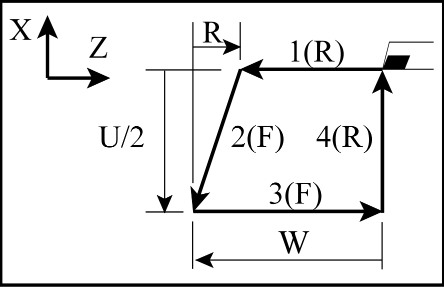
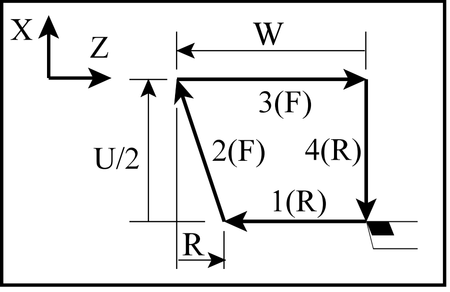
1. U<0, W<0, R<0	2. U>0, W<0, R<0
	
3. U<0, W<0, R>0, és |R|≤|W|	3. U>0, W<0, R>0, és |R|≤|W|
	

Fig. 18.1.3-3

18.1.4 Application of Single Cycles

Both the code G and the input parameters of the cycles are modal. This means, that if the variables X(Z, Y), U(W V), Z(Y, X), W(V, U), or R(I, J or K) are already given and their values are unchanged, they should not be written in the program again. For example:

```
G91 G18...
G77 X-20 Z-50 F0.5
X-30
X-40
X-50
G0... (cycle is deleted)
...
```

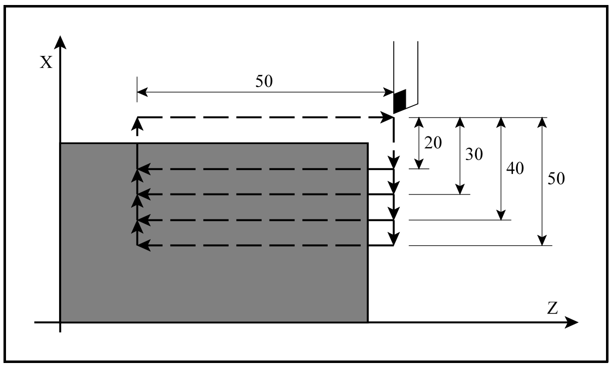


Fig. 18.1.4-1

In the example above, it is only the cutting depth the value of which (X) does not change, so only this address has to be filled, the value of the other ones remains unchanged.

In the switched-on state of the cycle, it will be executed in the only case if one of the motion-related variables X(U), Y(V), Z(W) is also filled. For example, if in cycle state a function is being executed in a separate block, the cycle state will remain switched on, but the cycle will not be repeated:

```
G18...
G77 U-20 W-50 F0.5  (it switches on and executes the cycle)
T202                (the cycle is switched on, but it does
                    not executes it)
U-30                (it executes the cycle)
G1...               (the cycle is deleted)
...
```

The interpolation codes G belonging to the group 1 are those that delete the cycle and the modal variables.

Function M, S and T can also be written in the blocks containing single cycles. Functions are always executed in the action 1 of the cycle, either parallelly with the motion or at the end of the motion. If, in certain cases, it is not good, the function will have to be written in a separate block.

18.2 Multiple Repetitive Cycles

The multiple repetitive cycles simplify the writing of part program. For example, the contour of the finished workpiece has to be described for finishing. This contour at the same time, determines the basis of the cycles of workpiece roughing (G71, G72 and G73). In addition to the roughing cycles, a finishing cycle (G70), a threading cycle (G76) and two grooving cycles (G74 and G75) are also available.

18.2.1 Roughing Cycle (G71)

Two types of roughing cycle can be distinguished: the type 1 and the type 2.

Type 1 Roughing Cycle

If the finished workpiece contour marked in the figure with the points A–A'–B is given, the roughing of the unmachined workpiece will be executed by the cycle G71 with depth of cut Δd and leaving the finishing allowances Δu/2 and Δw.

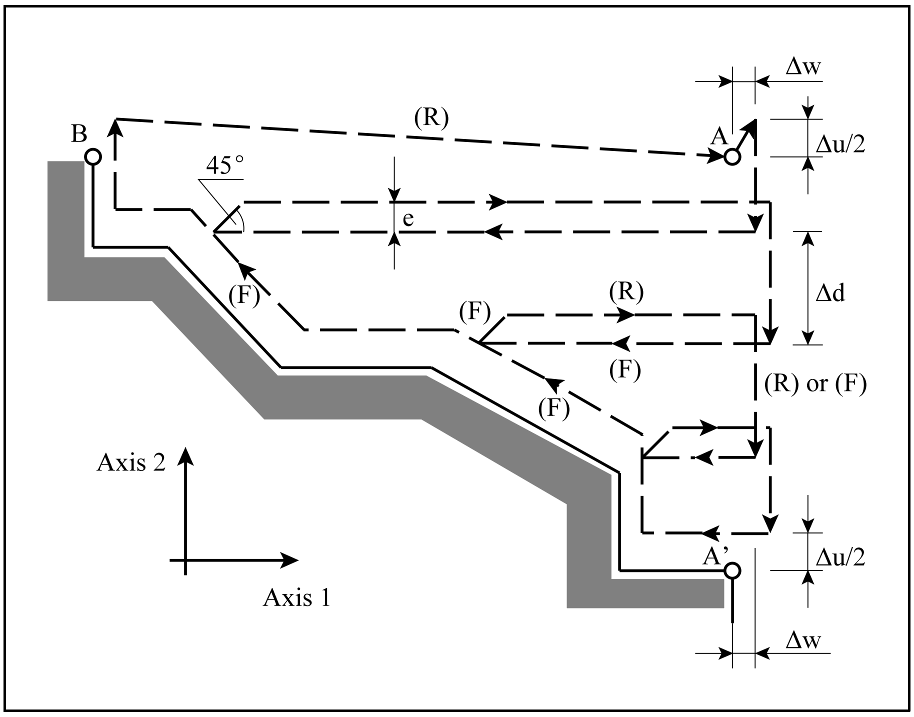


Fig. 18.2.1-1

Method 1 of specification:
Specification of the roughing cycle is done by filling the parameters of two succeeding blocks.

G18 (plane Z-X)
  G71 **U**(Δd) **R** (e)
  G71 **P** (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) X(U) ...
    ...
    N(n_f)...

G19 (plane Y-Z)
  G71 **W**(Δd) **R** (e)
  G71 **P** (n_s) **Q** (n_f) **W**(Δu) **V**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) Z(W) ...
    ...
    N(n_f)...

G17 (plane X-Y)
  G71 **V**(Δd) **R** (e)
  G71 **P** (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) Y(V) ...
    ...
    N(n_f)...

where:

Δd: depth of cut. It is a ***positive*** number to be always interpreted in ***radius***. If the value of the depth of cut is not given in the program, the control will have it from the parameter N1600 Depth of Cut.

e: escaping. It is a ***positive*** number to be always interpreted in ***radius***. If the value of the escaping is not given in the program, the control will have it from the parameter N1601 Escape.

n_s: starting block number of the program part describing the finishing (A–A'–B course).

n_f: final block number of the program part describing the finishing (A–A'–B course).

Δu: value and direction of the finishing allowance ***along the second axis of the selected plane***. It is a ***signed*** number to be interpreted in ***diameter or radius***, depending on interpretation of the second axis.

Δw: value and direction of the finishing allowance ***along the first axis of the selected plane***. It is a ***signed*** number to be interpreted in ***diameter or radius***, depending on interpretation of the first axis.

f, **s**, **t**: in the course of the cycle, the F, S and T functions programmed in the program part from n_s to n_f describing the finishing (A–A'–B course) are ignored by the control, but the control validates the values f, s and t given in the block G71 or which are modal.

The value given at the address U (W, V) can mean Δd or Δu depending on whether P and Q are programmed or not. If not, the address U (W, V) will mean Δd; if yes, the address U (W, V) will mean Δu.
The roughing cycle will be executed by the block in which P and Q are given.

Motion between the points ***A–A'*** must be specified in the block given at the address P and numbered by n_s , by ***obligatory programming the G0 or G1***. The code given here determines whether, in the course of roughing, ***infeed*** (motion in the direction A–A') is executed ***at rapid traverse rate*** (in the case of programming the ***G0***) or ***at feed rate*** (in the case of programming the ***G1***). In this block P(n_s), the following motions must always be specified: in the case of G18 - ***motion in direction X***, in the case of G19 - motion in direction Z, and in the case of G17 - motion in direction Y; referring to other axis is not allowed.

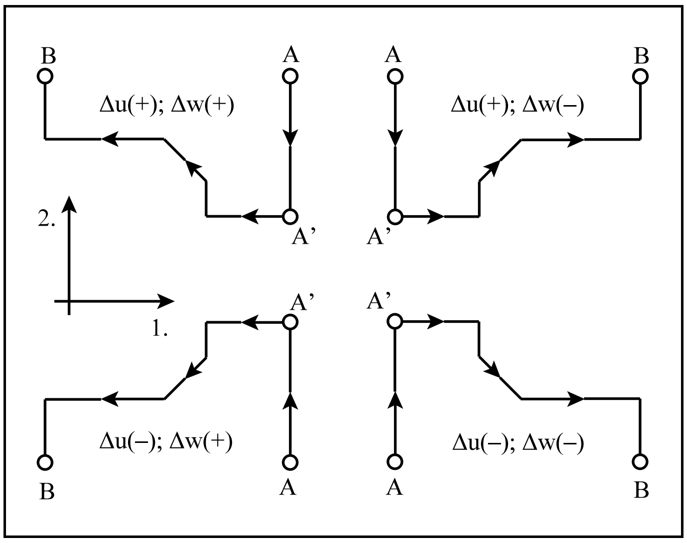


Fig. 18.2.1-2

The course A'–B is the real contour composed of straight lines and circle arcs. ***In the case of type 1 roughing cycle***, ***the contour has to be monotonously increasing or decreasing*** both in the directions X (Z, Y) and Z (Y, X), which means that ***regression is not possible in any directions***. The cycle can be used in all the four plane quadrants. In the figure the sign of the finish allowance is also shown.

The control ***ignores the F, S and T functions programmed in the program part between the blocks n_s and n_f***, and it validates those that were programmed in the block G71 (f, s, t) or previously. This also concerns the constant surface speed programmed between the blocks n_s and n_f, i.e. the control validates ***the state G96 or G97, and constant surface speed that were valid before the block G71***.

Subprogram call in the blocks from n_s to n_f is not allowed.

Calculation of tool nose radius compensation (G41 and G42) can be switched on in the course of cycle execution with the restriction that the switch on (G41 or G42) and the switch off (G40) has to be done between the blocks n_s and n_f:

CORRECT

```
N(ns)  X(U)  G41 ...
        (G41)...
        ...
       (G40)
N(nf)  G40 ...
```

or

```
       G41
N(ns)  X(U)  ...
        ...
        ...
N(nf)  ...
       G40
```

WRONG

```
       G41
N(ns)  X(U)  ...
       ...
       ...
      G40
N(nf)  ...
```

or

```
N(ns)  G41  X(U)  ...
        ...
        ...
N(nf)  ...
       G40
```

Method 2 of specification:

G18 (plane Z-X)

G71 **P** (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)

N(n_s) X(U) ...

...

N(n_f)...

G19 (plane Y-Z)

G71 **P** (n_s) **Q** (n_f) **W**(Δu) **V**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)

N(n_s) Z(W) ...

...

N(n_f)...

G17 (plane X-Y)

G71 **P** (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)

N(n_s) Y(V) ...

...

N(n_f)...

Input parameters of both methods are the same.

Type 2 Roughing Cycle

The way of specification of the type 2 roughing cycle is the same as it was in the case of the type 1 roughing cycle; its code is G71, and it has the same input parameters .

The difference is in specification of the starting block of the contour (the block numbered by n_s). While in the case of calling the type 1 roughing cycle it is not allowed to refer to the first axis, the address Z (Y, X) in this block, i.e. the motion of the course A–A' must be perpendicular to the first axis Z (Y, X); in the case of calling the type 2 roughing cycle it is obligatory ***to refer to the address Z (Y, X)***. So, the course A–A' does not have to be perpendicular to the first axis Z (Y, X).

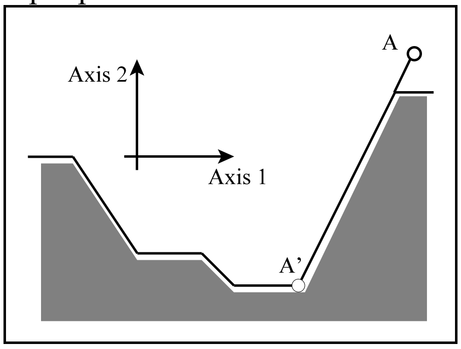


Fig. 18.2.1-3

Specification of the type 1

```
G18
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
N100 X(U)___
...
...
...
N200
```

Specification of the type 2

```
G18
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
N100 X(U)___ Z(W)__
...
...
...
N200
```

In the case, when type 2 cycle has to be used, but in the block starting the contour, motion has to be executed in the direction X (Z, Y) only, i.e. perpendicularly to the axis Z (Y, X), incremental displacement 0 along the axis Z (Y, X) has to be programmed, i.e. Z (Y, X) I0 or W (V, U).

In the case of type 2 roughing cycle, the contour does not have to be monotonously increasing or decreasing in the direction of the second axis X (Z, Y), i.e. the contour can be regressive and can have pockets. ***Programming the finishing allowance (Δw) in the direction of the first axis*** is not allowed, W (V, U) has to be 0, otherwise the tool could cut into one of the sides of the pocket.

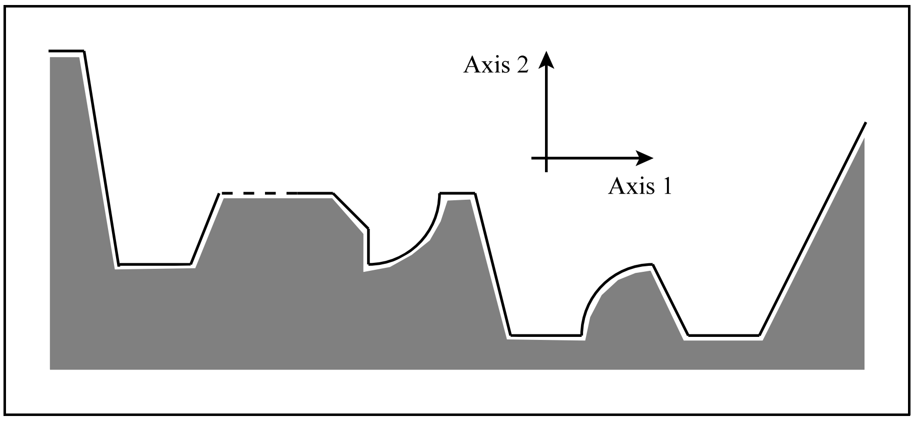


Fig. 18.2.1-4

However, ***in the direction of the first axis Z (Y, X)*** the contour ***has to remain monotonic***, it cannot be regressive.

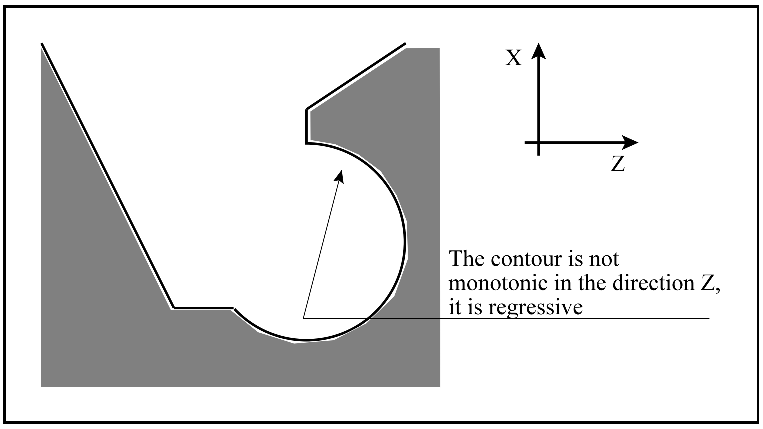

X
Z
The contour is not monotonic in the direction Z, it is regressive

Fig. 18.2.1-5

In the case of the type 2 roughing cycle, the escaping is executed perpendicularly to the axis Z with a valid escaping value e.

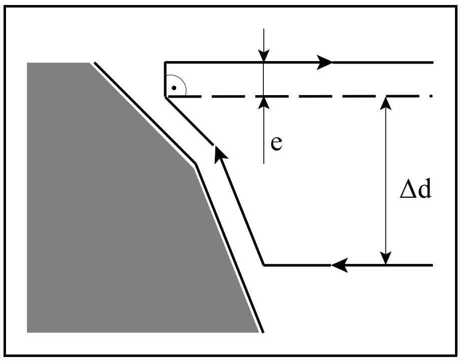


Fig. 18.2.1-6

According to the bit position #2 FPT of the parameter N1611 Turning Cyc. Config., the type 2 roughing cycle can cut the pockets in two different ways. If FPT:

=0: the cycle will be started by cutting ***the last pocket in the direction of contour following***;

=1: the cycle will be started by cutting ***the first pocket in the direction of contour following***.

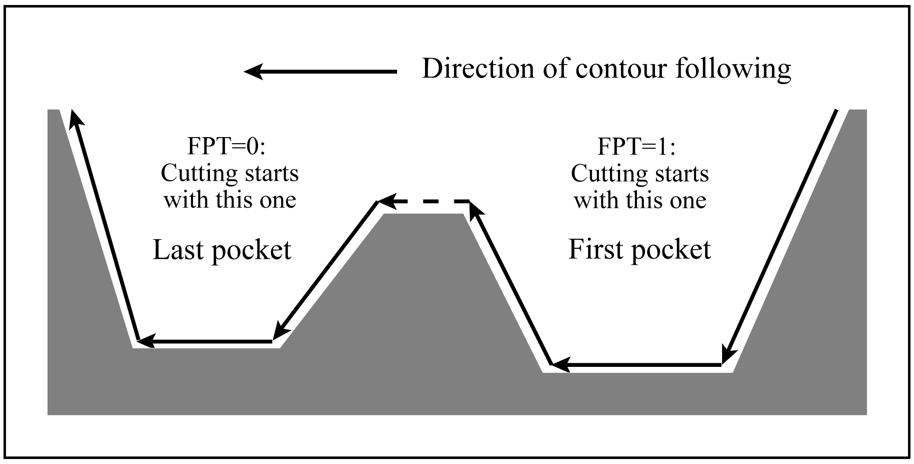

Direction of contour following
FPT=0: Cutting starts with this one
Last pocket
FPT=1: Cutting starts with this one
First pocket

Fig. 18.2.1-7

The following applies both to the type 1 and type 2 roughing cycles.

Continuing the program after execution of a roughing cycle

After execution of a roughing cycle, machining will continue either by execution of ***the blocks succeeding the block G71 P Q*** or ***after the block having the number specified at the address Q***. In the first case, if the starting block of the contour succeeds the block G71 P Q, the execution will move to the contour following, and finishing of the workpiece will be done.
In the second case, the program execution will continue by the block succeeding the contour final block specified at the address Q, and for this reason finishing cycle G70 has to be programmed for finishing. It is useful in the case, when several roughing or facing cycles are carried out by the use of the same roughing tool, and then having changed a finishing tool these several will be finished using the finishing cycle G70.
Selection between the two options above can be made on the basis of the bit #1 SKP of the parameter N1611 Turning Cyc. Config.. If SKP:

=0: the program will continue by the block succeeding the command G71, G72, G73;
=1: the program will continue by the block succeeding the block specified at the address Q in the command G71, G72, G73.

SKP=0

```
G18
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
(the program continues here)
N100
...
N200
(finishing is completed)
```

SKP=1

```
G18
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
N100
...
N200
(the program continues here)
...
G70 P100 Q200 (finishing)
```

Path monotonicity test

Both in type 1 and type 2 roughing cycles, ***the contour must be monotonic along the roughing axis***, i.e. regressive element is not allowed along the path. If the path is not monotonic, the control will send the error message ‘The contour is not monotonic’.
It is possible to set a tolerance for monotonicity in the parameter N1613 Tolerance Along Roughing Axis which will be taken into account by the control for the cycles G71 and G72, too. In the case, when the degree of regression is smaller than the value given in the parameter, the control will not send an error message.

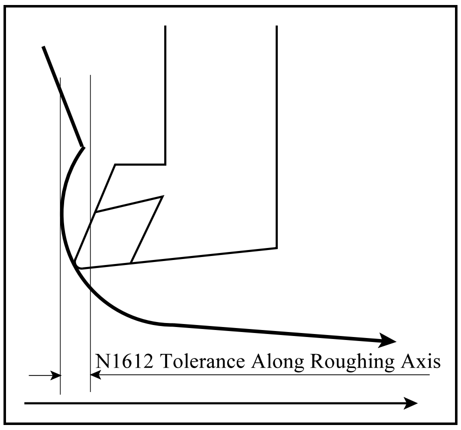


Fig. 18.2.1-8

The type 1 roughing cycle ***must*** also ***be monotonic along the cutting*** (infeed) ***axis***, i.e. pocket is not allowed along the path. If the path is not monotonic, the control will send the error message 'The contour is not monotonic'.
It is possible to set a tolerance for monotonicity in the parameter N1614 Tolerance Along Cutting Axis which will be taken into account by the control for the type 1 cycles G71 and G72, too. In the case, when the degree of regression is smaller than the value given in the parameter, the control will not send an error message.

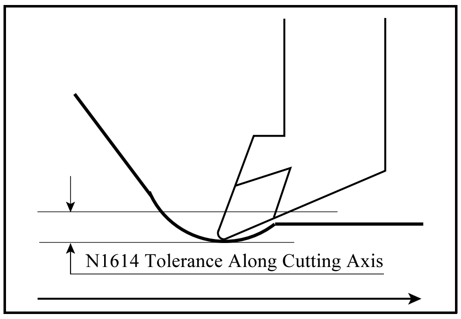


Fig. 18.2.1-9

Both in type 1 and type 2 roughing cycles, the path can be regressive ***in the moment of switching on*** (G41, G42) and ***switching off*** (G40) ***the tool nose radius compensation*** even in the case, when the programmed contour is monotonic.
The reason for this is that in the state G40 the imaginary tip of the tool is guided by the control along the path, while in the state G41, G42 the tool nose circle is guided parallelly with the contour.
There can be set ***the monotonicity test of the contour*** passed between the blocks numbered at the address P and Q in the roughing cycles G71, G72; using the bit #3 FCK of the parameter N1611 Turning Cyc. Config. . If the value of the parameter:

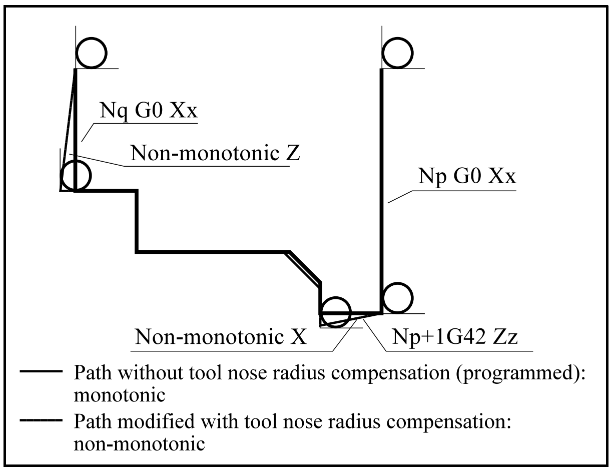

Nq G0 Xx
Non-monotonic Z
Np G0 Xx
Non-monotonic X
Np+1G42 Zz
Path without tool nose radius compensation (programmed): monotonic
Path modified with tool nose radius compensation: non-monotonic

Fig. 18.2.1-10

=0: ***the test is carried out for the path modified with radius compensation;***
=1: ***the test is carried out for the original path not modified with radius compensation***.

Examples
The following bit states of the parameter N1611 Turning Cyc. Config. are assumed in the examples below:
#1 SKP=1
#3 FCK=1

An example for the type 1 cycle G71:

```
G18...
N10 G54 G0 X200 Z50
N20 G92 S3000
N30 G96 S400
N40 T101 M3 (ROUGHING TOOL, Q3)
N50 G0 X160 Z2
/N60 G71 U4 R1
/N70 G71 P80 Q160 U1 W0.5 F0.5 (D4) (ROUGHING)
N80 G0 X100
N90 G1 G42 Z0
```

```
N100 X110 ,C2
N110 Z-25
N120 G2 X130 Z-35 R10
N130 G1 X140 ,R2
N140 Z-45 ,C3
N150 X155
N160 G40 X160
N170 G0 X200 Z50
N180 T202 (FINISHING TOOL, Q3)
N190 G0 X160 Z2
/N200 G70 P80 Q160 (FINISHING)
N210 G0 X200 Z50
...
```

An example for the type 2 cycle G71:

```
G18...
N10 G54 G0 X200 Z50
N20 G92 S3000
N30 G96 S400
N40 T606 M3 (ROUGHING TOOL, Q8)
N50 G0 X180 Z2
/N60 G71 U6 R4
/N70 G71 P80 Q210 U1 (D6) (ROUGHING)
N80 G42 G0 X98 Z2
N90 G1 Z-10
N100 X100
N110 ,A150 (intersection point between N110-N120)
N120 X110 Z-40 ,A225
N130 Z-50
N140 ,A150 (intersection point between N140-N150)
N150 G3 X110 Z-110 I55 K-80 R30 Q1      (intersection point
                                         between N150-N160)
N160 G1 X100 Z-130 ,A210 Q1   (intersection point between
                              N160-N170)
N170 G2 X100 Z-150 R10 Q-1    (intersection point between
                              N170-N180)
N180 X110 Z-180 I55 K-160 R20 Q-1
N190 G1 X170
N200 Z-190
N210 G40 G0 X180
N220 G0 X200 Z50
N230 T707 (FINISHING TOOL Q8)
/N240 G70 P80 Q210 (FINISHING)
N250 G0 X200 Z50
...
```

Type 3 Roughing Cycle

The type 3 roughing cycle can be used when the recurving contour, which is not monotonous in the X-axis direction, is to be produced with two different tools. In such a case, ***two G71 calls*** must be programmed for the two different tools, but ***the contour description is the same for both G71s***, i.e. the contour description does not need to be split into two parts.
In such case, ***the first cycle produces the non-recurving part of the contour*** by the use of the first tool, and then, after changing over the second tool, ***the second cycle produces the recurving part of the contour***, i.e. the pockets.

Programming the Type 3 Roughing Cycle

The type 3 roughing cycle must be programmed in the same way as the type 2. The only difference is that type 3 cycle is identified by the H address.

G71 ... H

Interpretation of the H address:

H1: if the value of H is positive, the only non-recurving parts will be produced by the cycle,

H-1: if the value of H is negative, the only recurving parts will be produced by the cycle along the contour.

In the program, in both G71s, the same initial (P address) and completion (Q address) block numbers will be specified for the blocks descripting the contour:

```
G71 Pp Qq ... H1 (producing the non-recurving parts)
...
G71 Pp Qq ...H-1 (producing the recurving parts)
...
Np X Z (initial block of the contour)
...
Nq (completion block of the contour)
```

A sample program of the application of the Type 3 Roughing Cycle

Attention: In the following sample program, the #1 SKP=0 state of the parameter N1611 Turning Cyc. Config. is assumed!

```
...
N90 G54 G0 X200 Z50
N100 G92 S3000
N110 G96 S400
N120 T303 M4 (roughing tool, Q3)
N130 G0 X180 Z2
/N140 G71 U6 R4
/N150 G71 P240 Q370 U3 H1 (roughing the monotonous parts)
N160 G0 X200 Z50
N170 T606 M4 (roughing tool, Q8)
/N180 G71 P240 Q370 U3 H-1 (roughing the pockets)
N190 G0 X200 Z50
N200 T101 M4 (finishing tool, Q8)
N210 G70 P240 Q370 (finishing )
N220 Z50
N230 M30 (end of the program)
N240 G42 G0 X98 Z2 (start of the contour)
N250 G1 Z-10
N260X100
N270 ,A150
N280 X110 Z-40 ,A225
N290 Z-50
N300 ,A150 (X161.962; Z-95.)
N310 G3 X110 Z-110 I55 K-80 R30 Q1
N320 G1 X100 Z-130 ,A210 Q1
N330 G2 X100 Z-150 R10 Q-1
N340 X110 Z-180 I55 K-160 R20 Q-1
N350 G1 X170
N360 Z-190
N370 G40 G0 X180 (end of the contour)
```

18.2.2 Face Roughing Cycle (G72)

The face roughing cycle (G72) is the same as the roughing cycle G71 with the exception that cutting is executed parallelly with the second axis of the plane. ***Everything described for the cycle G71 are also valid for the cycle G72***, so they are not detailed herein.

Type 1 Face Roughing Cycle

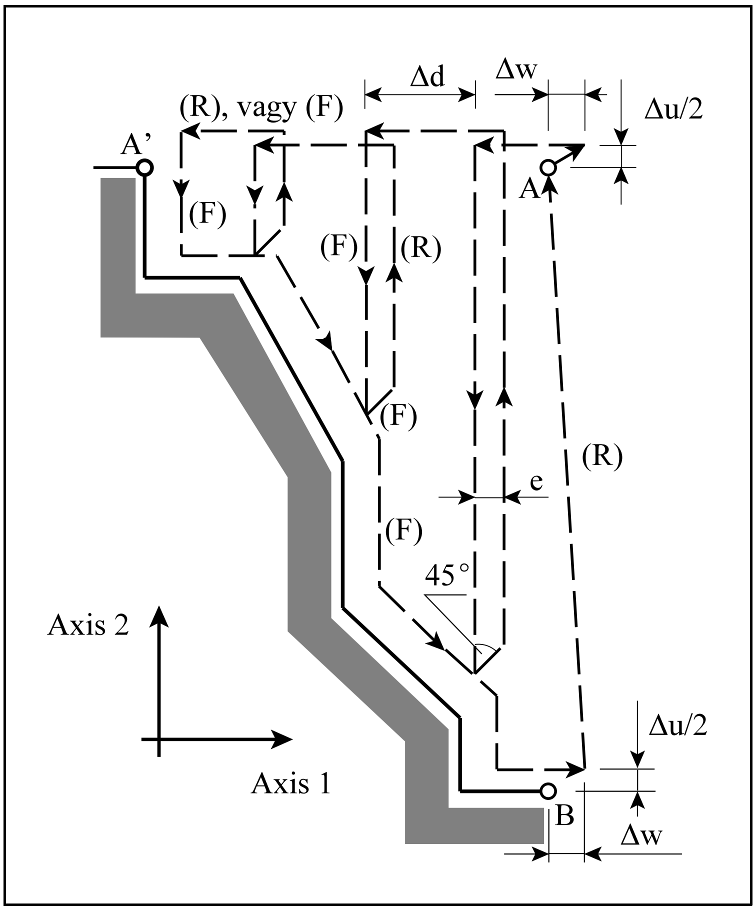


Fig. 18.2.2-1

Method 1 of specification:

G18 (plane Z-X)
  G72 **W**(Δd) **R** (e)
  G72 **P** (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) Z(W) ...
    ...
    N(n_f)...

G19 (plane Y-Z)
  G72 **V**(Δd) **R** (e)
  G72 **P** (n_s) **Q** (n_f) **W**(Δu) **V**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) Y(V) ...
    ...
    N(n_f)...

G78 (plane X-Y)
  G72 **U**(Δd) **R** (e)
  G72 **P** (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **F**(f) **S**(s) **T**(t)
    N(n_s) X(U) ...
    ...
    N(n_f)...

Meaning of the parameters is the same as it was in the case of the cycle G71.

Method 2 of specification:

G18 (plane Z-X)
  G72 **P** (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)
    N(n_s) Z(W) ...
    ...
    N(n_f)...

G19 (plane Y-Z)
  G72 **P** (n_s) **Q** (n_f) **W**(Δu) **U**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)
    N(n_s) Y(V) ...
    ...
    N(n_f)...

G78 (plane X-Y)
  G72 **P** (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)
    N(n_s) X(U) ...
    ...
    N(n_f)...

Motion between the points ***A–A'*** must be specified in the block given at the address P and numbered by n_s. In this block P(n_s), the following motions must always be specified: in the case of G18 - ***motion in direction Z***, in the case of G19 - motion in direction Y, and in the case of G17 - motion in direction X; referring to other axis is not allowed.

The cycle can be used in all the four plane quadrants. In the figure the sign of the finish allowance is also shown for all the four cases.

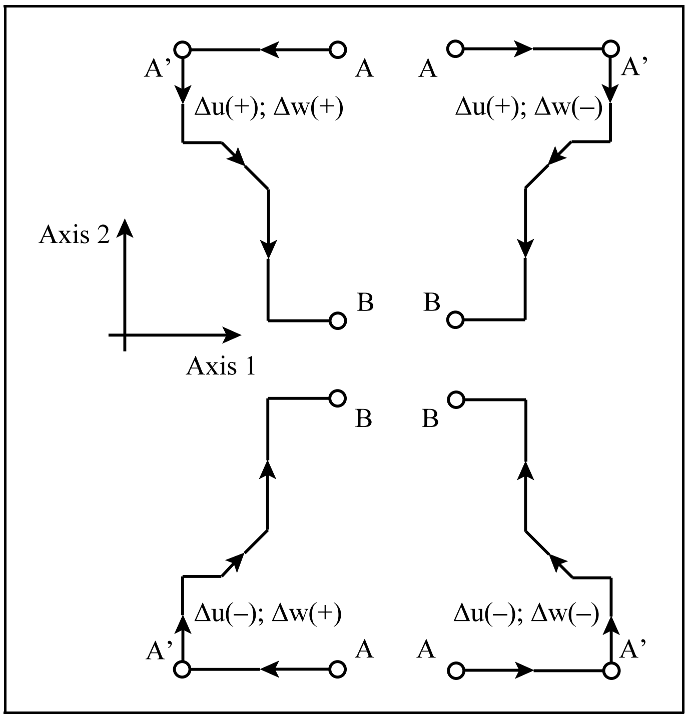


Fig. 18.2.2-2

Type 1 Face Roughing Cycle

Specification of the type 2 face roughing cycle should be done in the same way as it was done in the case of the type 1 face roughing cycle, its code is G72, and its input parameters are the same, too.

The difference is in specification of the starting block of the contour (the block numbered by n_s). While in the case of calling the type 1 face roughing cycle it is not allowed to refer to the second axis, the address X (Z, Y) in this block, i.e. the motion of the course A–A' must be perpendicular to the second axis X (Z, Y) ; in the case of calling the type 2 face roughing cycle it is obligatory ***to refer to the address X (Z, Y)***. So, the course A–A' does not have to be perpendicular to the second axis X (Z, Y).

Examples

The following bit states of the parameter N1611 Turning Cyc. are assumed in the examples below:

#1 SKP=1
#3 FCK=1

An example for the type 1 cycle G72:

```
G18...
N10 G54 G0 X200 Z50
N20 G92 S3000
N30 G96 S600
```

```
N40 T101 M3 (ROUGHING TOOL, Q3)
N50 G0 X161 Z2
/N60 G72 W5 R1
/N70 G72 P80 Q140 U1 W0.5 (D5) F0.5 (ROUGHING)
N80 G41 G0 Z-25
N90 G1 X120 ,C3
N100 Z-17 ,R2
N110 G3 X100 Z-7 R10
N120 G1 Z-3 ,C1
N130 X80
N140 G40 Z2
N150 G0 X200 Z50
N160 T202 (FINISHING TOOL Q3)
N170 X161 Z2
/N180 G70 P80 Q140 (FINISHING)
N190 G0 X200 Z50
...
```

An example for the type 2 cycle G72:

```
G18...
N10 T505 (ROUGHING TOOL, Q7)
N20 G54 G0 X200 Z50
N30 G92 S3000
N40 G96 S200  M3
N50 G0 X160 Z5
/N60 G72 W4 R1
/N70 G72 P80 Q150 W0.5 (D4) (ROUGHING)
N80 G41 G0 X160 Z-25
N90 G1 X145 ,R4
N100 X125 Z-10 ,R3
N110 X105  Z-25 ,R4
N120 X95
N130 G2 X15 Z-25 R20 ,R5
N140 G1 X0
N150 G40 Z5
N160 G0 X200 Z50
N170 T202 (FINISHING TOOL Q7)
N180 X160 Z5
/N190 G70 P80 Q150 (FINISHING)
N200 G0 X200 Z50
...
```

Type 3 Face Roughing Cycle

The type 3 face roughing cycle can be used when the recurving contour, which is not monotonous in the Z-axis direction, is to be produced with two different tools. In such a case, ***two G72 calls*** must be programmed for the two different tools, but ***the contour description is the same for both G72s***, i.e. the contour description does not need to be split into two parts.

In such case, ***the first cycle produces the non-recurving part of the contour*** by the use of the first tool, and then, after changing over the second tool, ***the second cycle produces the recurving part of the contour***, i.e. the pockets.

Programming the Type 3 Face Roughing Cycle

The type 3 face roughing cycle must be programmed in the same way as the type 2. The only difference is that type 3 cycle is identified by the H address.

G72 ... H

Interpretation of the H address:

H1: if the value of H is positive, the only non-recurving parts will be produced by the cycle,

H-1: if the value of H is negative, the only recurving parts will be produced by the cycle along the contour.

In the program, in both G72s, the same initial (P address) and completion (Q address) block numbers will be specified for the blocks descripting the contour:

```
G72 Pp Qq ... H1 (producing the non-recurving parts)
...
G72 Pp Qq ...H-1 (producing the recurving parts)
...
Np X Z (initial block of the contour)
...
Nq (completion block of the contour)
```

A sample program of the application of the Type 3 Face Roughing Cycle

Attention: In the following sample program, the #1 SKP=0 state of the parameter N1611 Turning Cyc. Config. is assumed!

```
...
N80 G54 G0 X200 Z50
N90 G92 S3000
N100 G96 S200
N110 T303 M4 (roughing tool, Q3)
N120 G0 X160 Z5
/N130 G72 W5 R3
/N140 G72 P230 Q300 W2 H1 (roughing the monotonous parts)
N150 G0 X200 Z50
N160 T404 S1=500 M4 (roughing tool, Q7)
/N170 G72 P230 Q300 W2 H-1 (roughing the pockets)
N180 G0 X200 Z50
N190 T505 S1=500 M4 (finishing tool, Q7)
N200 G70 P230 Q300 (finishing)
N210 G0 X200 Z50
N220 M30 (end of the program)
N230 G41 G0 X160 Z-25 (start of the contour)
N240 G1 X145 ,R4
N250 X125 Z-10 ,R3
N260 X105  Z-25 ,R4
N270 X95
N280 G2 X15 Z-25 R20 ,R5
N290 G1 X0 (Z-30)
N300 G40 (X-2) Z5 (end of the contour)
```

18.2.3 Pattern Repeating Cycle (G73)

This cycle can be used for roughing pre-forged, cast or pre-roughed workpieces having the outline of the final shape already. The cycle repeats pass by pass a contour written in the program.

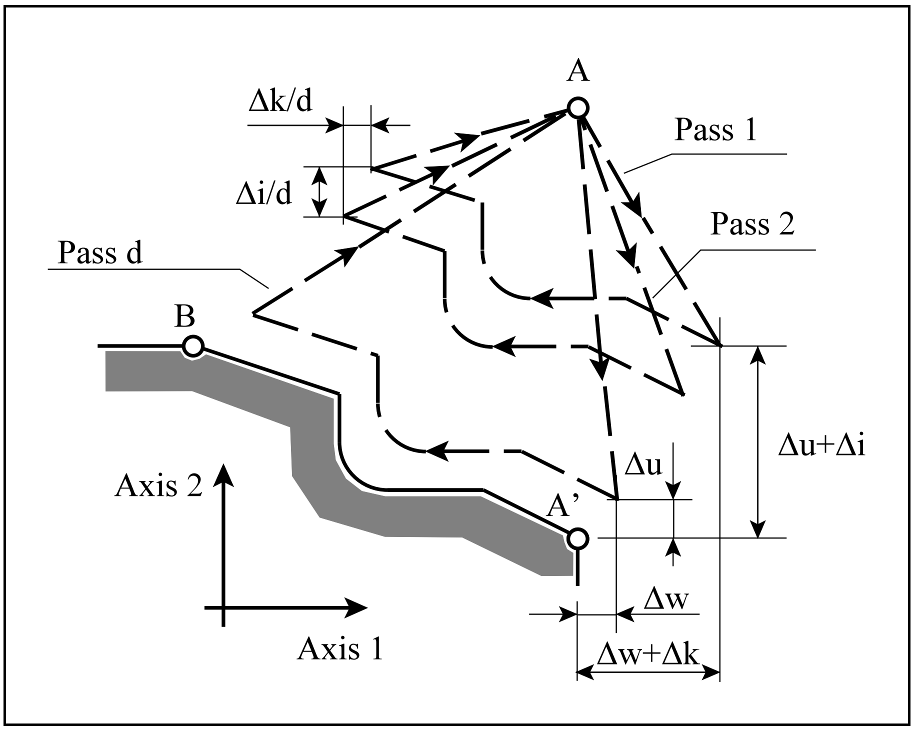


Fig. 18.2.3-1

Method 1 of specification:

G18 (plane Z-X)
G73 U(Δi) **W**(Δk) **R** (d)
G73 P (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

G19 (plane Y-Z)
G73 W(Δi) **V**(Δk) **R** (d)
G73 P (n_s) **Q** (n_f) **W**(Δu) **V**(Δw) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

G17 (plane X-Y)
G73 V(Δi) **U**(Δk) **R** (d)
G73 P (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

series of commands is used, where:

Δi: value and direction of the roughing allowance along the second axis of the selected plane. It is a ***signed*** number to be always interpreted in ***radius***. If the value of the roughing allowance is not given in the program, the control will have it from the parameter N1603 Relief Ax2.

Δk: value and direction of the roughing allowance along the first axis of the selected plane. It is a ***signed*** number to be always interpreted in ***radius***. If the value of the roughing allowance is not given in the program, the control will have it from the parameter N1602 Relief Ax1.

d: number of divisions. If the number of passes is not given in the program, the control will have it from the parameter N1604 Numb of Div.

n_s: starting block number of the program part describing the finishing (A–A'–B course).

n_f: final block number of the program part describing the finishing (A–A'–B course).

Δu: value and direction of the finishing allowance along the second axis of the selected plane. It is a ***signed*** number to be interpreted in ***diameter or radius***, depending on interpretation of the coordinate.

Δw: value and direction of the finishing allowance along the first axis of the selected plane. It is a ***signed*** number to be interpreted in ***diameter or radius***, depending on interpretation of the coordinate.

The addresses U (W, V) and W (V, U) given in the block G73 can mean Δi and Δk, or Δu and Δw depending on whether P and Q are programmed in the given block. In other words, if P and Q are programmed, U and W will mean Δu and Δw, respectively; if they are not programmed, U and W will mean Δi and Δk, respectively.
The cycle will be executed in the block containing P and Q. The cycle can be executed in all the four plane quadrants depending on the sign of the values Δi, Δk, Δu and Δw. After each pass and at the end of the cycle, the tool returns to the point 'A'.

The control ***ignores the F, S and T functions programmed in the program part between the blocks n_s and n_f***, and it validates those that were programmed in the block G73 (f, s, t) or previously. This also concerns the constant surface speed programmed between the blocks n_s and n_f, i.e. the control validates ***the state G96 or G97, and constant surface speed that were valid before the block G73***.
Subprogram call in the blocks from n_s to n_f is not allowed.
Calculation of tool nose radius compensation can be given in the blocks describing the cycle with the restrictions relating to the function G71.

Continuing the program after execution of a pattern repeating cycle

After execution the cycle, machining will continue either by execution of the blocks succeeding the block G73 P Q, or after the block numbered at the address Q. In the latter case, the workpiece can be finished using the finishing cycle G70.
Selection between the two options above can be made on the basis of the bit #1 SKP of the parameter N1611 Turning Cyc. Config.. If SKP:

=0: the program will continue by the block succeeding the command G71, G72, G73;

=1: the program will continue by the block succeeding the block specified at the address Q in the command G71, G72, G73.

Method 2 of specification:

G18 (plane Z-X)
G73 **P** (n_s) **Q** (n_f) **U**(Δu) **W**(Δw) **I**(Δi) **K**(Δk) **D**(d) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

G19 (plane Y-Z)
G73 **P** (n_s) **Q** (n_f) **W**(Δu) **V**(Δw) **K**(Δi) **J**(Δk) **D**(d) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

G17 (plane X-Y)
G73 **P** (n_s) **Q** (n_f) **V**(Δu) **U**(Δw) **J**(Δi) **I**(Δk) **D**(d) **F**(f) **S**(s) **T**(t)
N(n_s) ...
...
N(n_f) ...

Input parameters of both methods are the same.

Example

The bit state #1 SKP=1 of the parameter N1611 Turning Cyc. Config. are assumed in the examples below:

```
G18...
N10 G96 S200
N20 G92 S3000
N30 G54 G0 X70 Z20
N40 T111 M3(ROUGHING TOOL Q3)
/N50 G73 U10 W4 R4
/N60 G73 P70 Q130 U1 W0.5 (I10 K4 D4)
N70 G42 X10 Z5 F1.4
N80 G1 Z0
N90 X20 ,C1
N100 Z-10
N110 G2 X40 Z-20 R10
N120 G1 Z-30
N130 G40 G0 X42
N140 G0 X70 Z20
N150 T212 (FINISHING TOOL Q3)
/N160 G70 P70 Q130
N170 G0 X70 Z20
...
```

18.2.4 Finishing Cycle (G70)

After roughing the contour described by the blocks from n_s to n_f using the commands G71, G72 or G73, finishing the contour can be executed using the command G70. Finishing can be given by the command

G70 P (n_s) **Q** (n_f)

where

n_s:	starting block number of the program part describing the finishing ;
n_f:	final block number of the program part describing the finishing.

During the cycle, the ***functions F, S and T*** programmed in the program part from n_s to n_f and the ***commands*** related to the ***calculation of the constant cutting speed*** will be executed in contrast with the cycles G71, G72 and G73.
At the end of the finishing cycle, the block following the G70 will be read in.
In the course of the finishing cycle, calculation of ***tool nose radius compensation will be going on***.
Subprogram call is not allowed in the blocks from n_s to n_f.

Examples for the use of finishing cycles can be found at the cycles G71, G72 and G73.

18.2.5 Face Grooving Cycle (G74)

The figure below illustrates motion of the G74-type face grooving cycle. The grooving is executed along the first axis of the selected plane.

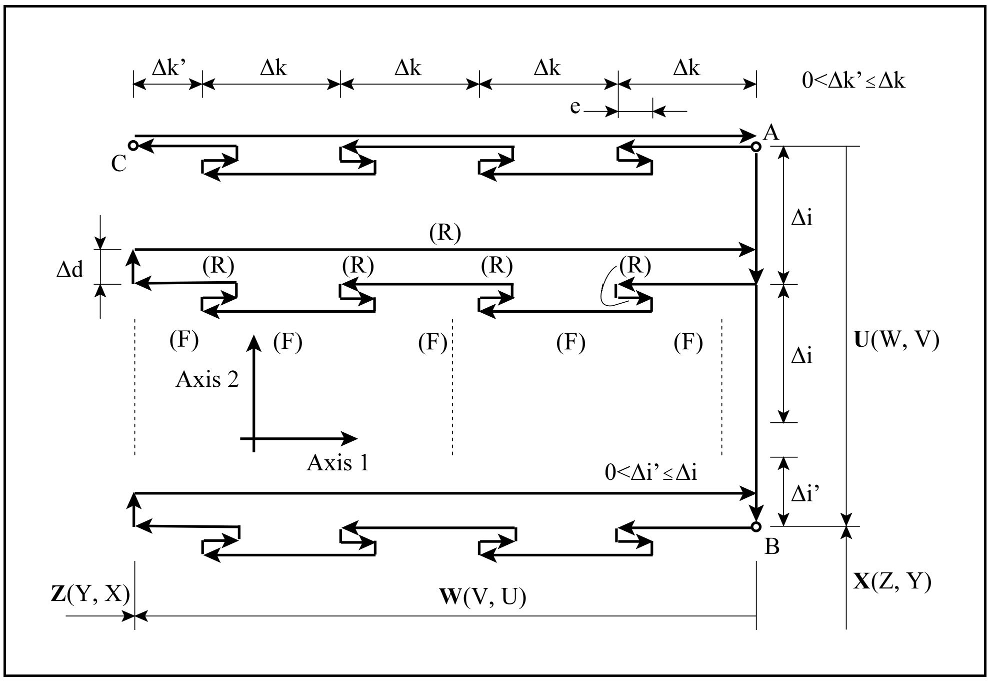


Fig. 18.2.5-1

Method 1 of specification:

G18 (plane Z-X)
G74 R (e)
G74 X$_p$(U) Z$_p$(W) P (Δi) **Q** (Δk) **R** (Δd) **F**

G19 (plane Y-Z)
G74 R (e)
G74 Y$_p$(V) Z$_p$(W) P (Δi) **Q** (Δk) **R** (Δd) **F**

G17 (plane X-Y)
G74 R (e)
G74 X$_p$(U) Y$_p$(V) P (Δi) **Q** (Δk) **R** (Δd) **F**

series of commands is used, where:

e: escaping. It is a modal ***positive number given in radius.*** It remains unchanged until it is rewritten. If the value of the escaping is not given in the program, the control will have it from the parameter N1605 Retr G74 G75.

X(Z, Y): absolute coordinate of the point 'B' given along the second axis of the plane.

U(W, V): incremental distance from the point A to the point B.

Z(Y, X): absolute coordinate of the point ‘C’ given along the first axis of the plane.

W(V, U): incremental distance from the point A to the point C.

Δi: infeed along the second axis of the plane. It is a positive number ***given in radius***.

Δk: infeed along the first axis of the plane. It is a positive number ***given in radius***.

Δd: relief of the tool at the cutting bottom. It is a number ***given in radius***. The motion direction is always contrary to the sign of the vector AB. At the end of the first grooving the displacement Δd is omitted.

F: feed.

In the figure, the distances covered at feed rate are marked with the letter F, and the distances covered at rapid traverse rate are marked with the letter R.
Filling the address Z(Y, X) or W(V, U) will determines whether filling the address R in the block G74 defines e or Δd. If the address is filled, the address R will mean Δd.
If filling the address X(Z, Y) or U(W, V) is omitted, and filling the address P(Δi) is also emitted, then motion will be executed along the axis Z only, i.e. a drilling cycle will occur.

Method 2 of specification:

G18 (plane Z-X)
G74 X_p(U) Z_p(W) I (Δi) **K** (Δk) **D** (Δd) **F**

G19 (plane Y-Z)
G74 Y_p(V) Z_p(W) K (Δi) **J** (Δk) **D** (Δd) **F**

G17 (plane X-Y)
G74 X_p(U) Y_p(V) J (Δi) **I** (Δk) **D** (Δd) **F**

The input parameters of the method 2 of specification have to interpreted as those of the method 1 of specification.

Example

```
G18...
G0 X100 Z5 S1=1500 M3 F1
G74 R6
G74 X70 Z-100 P6 (I6) Q25 (K25) R2 (D2) F1
G0 X120 Z10
...
```

18.2.6 Grooving Cycle (G75)

The figure below illustrates motion of the G75-type grooving cycle. The grooving is executed along the second axis of the selected plane.

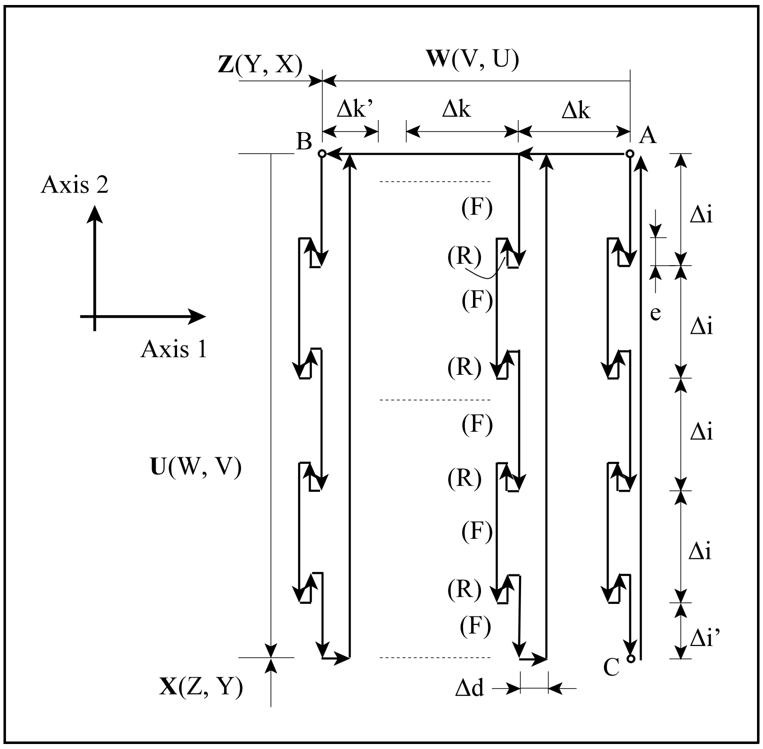


Fig. 18.2.6-1

Method 1 of specification:

G18 (plane Z-X)

G75 R (e)

G75 X_p(U) Z_p(W) P (Δi) **Q** (Δk) **R** (Δd) **F**

G19 (plane Y-Z)

G75 R (e)

G75 Y_p(V) Z_p(W) P (Δi) **Q** (Δk) **R** (Δd) **F**

G17 (plane X-Y)

G75 R (e)

G75 X_p(U) Y_p(V) P (Δi) **Q** (Δk) **R** (Δd) **F**

Variables of the cycle have to be interpreted as those of the cycle G74 with the exception that the grooving is executed in the direction X (Z, Y); for this reason, interpretation of the addresses X(Z, Y) U(W, V) and Z(Y, X) W(V, U) interchanges.

Method 2 of specification:

G18 (plane Z-X)
G75 X$_p$(U) Z$_p$(W) I (Δi) **K** (Δk) **D** (Δd) **F**

G19 (plane Y-Z)
G75 Y$_p$(V) Z$_p$(W) K (Δi) **J** (Δk) **D** (Δd) **F**

G17 (plane X-Y)
G75 X$_p$(U) Y$_p$(V) J (Δi) **I** (Δk) **D** (Δd) **F**

The input parameters of the method 2 of specification have to interpreted as those of the method 1 of specification.

Example

```
G18 ...
G0 X105 Z-15 S1=1500 M3 F1
G75 R6
G75 X20 Z-30 P15 (I15) Q6 (K6) R2 (D2) F1
G0 X120 Z10
...
```

18.2.7 Multiple Threading Cycle (G76)

The figure below illustrates motion of the G76-type multiple threading cycle.

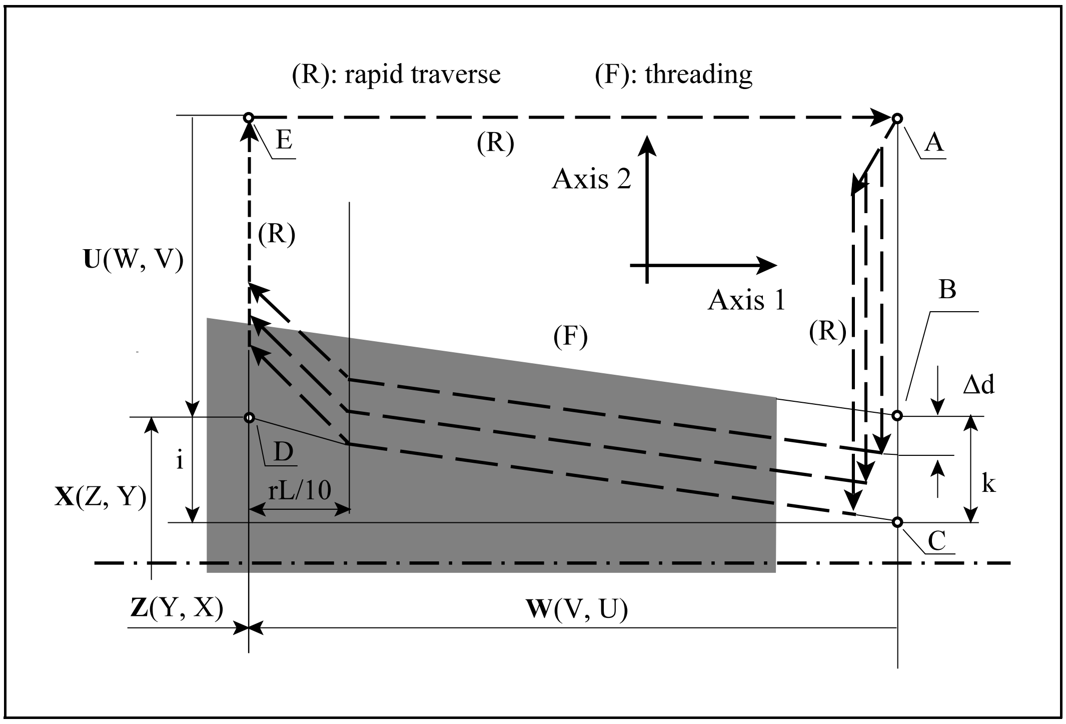


Fig. 18.2.7-1

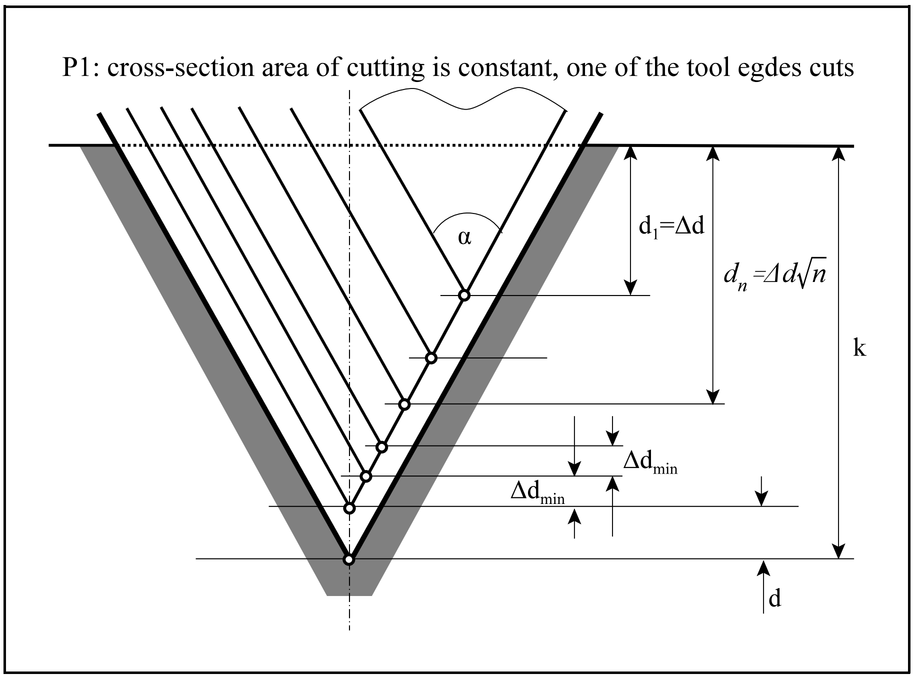

P1: cross-section area of cutting is constant, one of the tool egdes cuts
α
$d_1=\Delta d$
$d_n=\Delta d\sqrt{n}$
k
Δd_{min}
Δd_{min}
d

Fig. 18.2.7-2

Method 1 of specification:

G18 (plane Z-X)
G76 P (n) (r) (α) **Q** (Δd_{min}) **R** (d)
G76 $\mathbf{X_p}$(**U**) $\mathbf{Z_p}$(**W**) **P** (k) **Q** (Δd) **R** (i) **F**(**E**)(L)

G19 (plane Y-Z)
G76 P (n) (r) (α) **Q** (Δd_{min}) **R** (d)
G76 $\mathbf{Y_p}$(**V**) $\mathbf{Z_p}$(**W**) **P** (k) **Q** (Δd) **R** (i) **F**(**E**)(L)

G17 (plane X-Y)
G76 P (n) (r) (α) **Q** (Δd_{min}) **R** (d)
G76 $\mathbf{X_p}$(**U**) $\mathbf{Y_p}$(**V**) **P** (k) **Q** (Δd) **R** (i) **F**(**E**)(L)

series of commands is used.

The following parameters are the input data of the first block **G76 P** (n) (r) (α) **Q** (Δd_{min}) **R** (d).

n: ***repetitive count in finishing*** (n=01...99)

This value is modal, and it remains unchanged until it is rewritten. If the repetitive count in finishing is not given in the program, the control will have it from the parameter N1608 Count Fin.

r: ***chamfering*** (r=01...99)
When leaving the thread, the control pulls out the tool at an angle given in the parameter N1607 Chmfr Ang. The 'r' is used to give the length of pull out chamfer depending on the lead. The length is
$r{\cdot}L/10$
where L is the programmed lead.
This value is modal, and it remains unchanged until it is rewritten. If the chamfer is not given in the program, the control will have it from the parameter N1606 ThrdChmfr.

α: ***angle of the tool nose*** in degree (α=01...99)
This value is modal, and it remains unchanged until it is rewritten. If the angle of the tool nose is not given in the program, the control will have it from the parameter N1612 Tool Tip Angle.

The values of n, r and α can be specified at the address P all at once. Since each of the values is a two-digit number, a 6-digit number should be written at the address P. For example, if the repetitive count in finishing is n=2, the chamfering is 1.5L (r=15) and the angle of the threading tool nose is 60°, the value of P will be **P021560**.

$\mathbf{\Delta d_{min}}$: ***minimum depth of cut*** (to be always interpreted ***in radius***; it is a ***positive*** number).
If, in the course of threading, in the cycle 'n' the value of the depth of cut is $d_n - d_{n-1} < \Delta d_{min}$, the value of the depth of cut will always be limited to the value Δd_{min} by the control.
This value is modal, and it remains unchanged until it is rewritten. If the minimum depth of cut is not given in the program, the control will have it from the parameter N1609 Min Thrd Cut.

d: ***Finishing allowance*** (to be always interpreted ***in radius***; it is a ***positive*** number).
This value is modal, and it remains unchanged until it is rewritten. If the finishing allowance is not given in the program, the control will have it from the parameter N1610 Fin Allow.

By the code G76, the control will receive the parameters listed above if neither the address X(Z, Y) U(W, V) nor the address Z(Y, X) W(V, U) is filled in the block G76.

The following parameters are the input data of the second block **G76 X**(**U**) **Z**(**W**) **R** (i) **P** (k) **Q** (Δd) **F**(**E**)(L).

X_p(Z_p, Y_p): absolute coordinate of the point 'D' given along the second axis of the plane.

U(W, V): incremental distance from the point A to the point D.

Z_p(Y_p, X_p): absolute coordinate of the point 'D' given along the first axis of the plane.

W(V, U): incremental distance from the point A to the point D.

i: ***amount of taper*** (to be always interpreted ***in radius***).
If i=0 or the address R is not filled, straight thread will be cut.

k: ***height of thread*** (to be always interpreted ***in radius***, it is a ***positive*** number).

Δd: ***depth of first cut*** (to be always interpreted ***in radius***, it is a ***positive*** number).

L: ***lead of thread.***
It has to be programmed in the way that was valid for G33. The value written at the ***address F*** indicates ***lead of thread***, while the value written at the ***address E*** indicates ***threads per inch***.

Cutting thread will be executed only by the block filled in the manner above, which means that the addresses X(Z, Y) U(W, V) and Z(Y, X) W(V, U) must be filled. If none of the coordinate addresses are filled, the block will be interpreted by the control as parameter setting block.

During execution of the cycle, ***infeed*** will be done using ***rapid traverse*** if the code ***G0*** is valid in the course of the cycle, or using modal ***feed*** if the code ***G1*** is valid in the course of the cycle. Feed motion is executed between the points C and D in accordance with the lead of thread L given at the address F (E). Other sections will be covered using rapid traverse.

The tread is always cut on one side in the manner illustrated in the figure according to the formula

$$d_n = \Delta d\sqrt{n}$$

so that the cross-section area of cutting is constant.

At the end of the thread, chamfer is always cut according to the preset parameters.

Direction of infeed and direction of threading are determined by the motion direction programmed at the addresses X(Z, Y) U(W, V), Z(Y, X) W(V, U), respectively; while direction of the taper is determined by the sign of the address R.

In case of pushing button STOP, the control retracts the tool in accordance with the programmed chamfering and then positions to the initial point A, as it was described at the code G78. When the button START is pushed, the control begins the interrupted infeed again.

Method 2 of specification:

G18 (plane Z-X)
G76 X$_p$(U) **Z**$_p$**(W) I**(i) **K**(k) **D**(Δd) **A**(α) **F**(**E**)(L) **Q P**

G19 (plane Y-Z)
G76 Y$_p$**(V) Z**$_p$**(W) K**(i) **J**(k) **D**(Δd) **A**(α) **F**(**E**)(L) **Q P**

G17 (plane X-Y)
G76 X$_p$**(U) Y**$_p$**(V) J**(i) **I**(k) **D**(Δd) **A**(α) **F**(**E**)(L) **Q P**

The block specification will be considered by the control to be being made according to the method 2 if ***the address K(I,J) is filled***.

The input parameters n, r, α, Δd_{min} és d are also taken into account by the control in the case of thread specification made by the method 2 as it is done by the method 1: either by the parameter setting block **G76 P** (n) (r) (α) **Q** (Δd_{min}) **R** (d) or from parameter.

At the address 'A', the angle of the threading tool nose can be given, similarly to the method 1. The difference is that while the angle can be given with the resolution of 1° using integer number in the case of the method 1, in the case of the method 2 fractional number can also be used for it. If the address 'A' is not filled, the control will have the value of α from the parameter N1612 Tool Tip Angle.

The data i, k, Δd, L have to be interpreted as it was described at the method 1 of specification. Interpretation of other addresses is as follows:

Q: ***angle of tread start counted from the zero pulse of the encoder given in degree.*** Interpretation of the address is the same as it was at the G33.

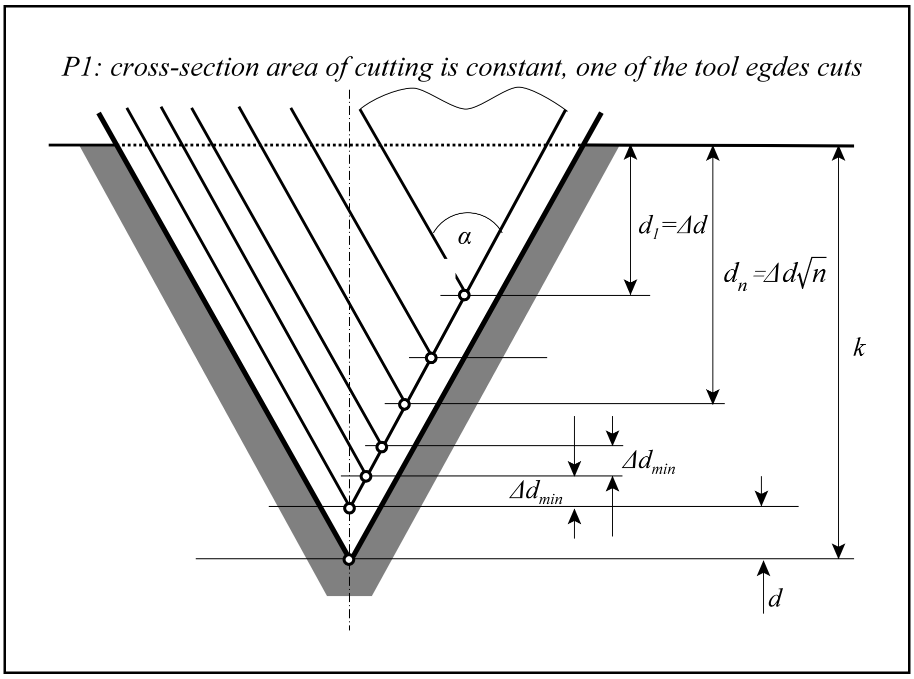
P: ***method of threading***.
The following four types of infeed illustrated in the figures below are available:
P1: cross-section area of cutting is constant, one of the tool edges cuts;
P2, P5: cross-section area of cutting is constant, both tool edges cut;
P3: infeed is constant, one of the tool edges cuts;
P4: infeed is constant, both tool edges cut.

Example

```
G18...
G97 S1000 M3
G0 X36 Z4
G76 P010560 Q0.3 R0.2
G76 X24 Z-50 P3 Q1 R-4 F3 (method 1 of specification)
```

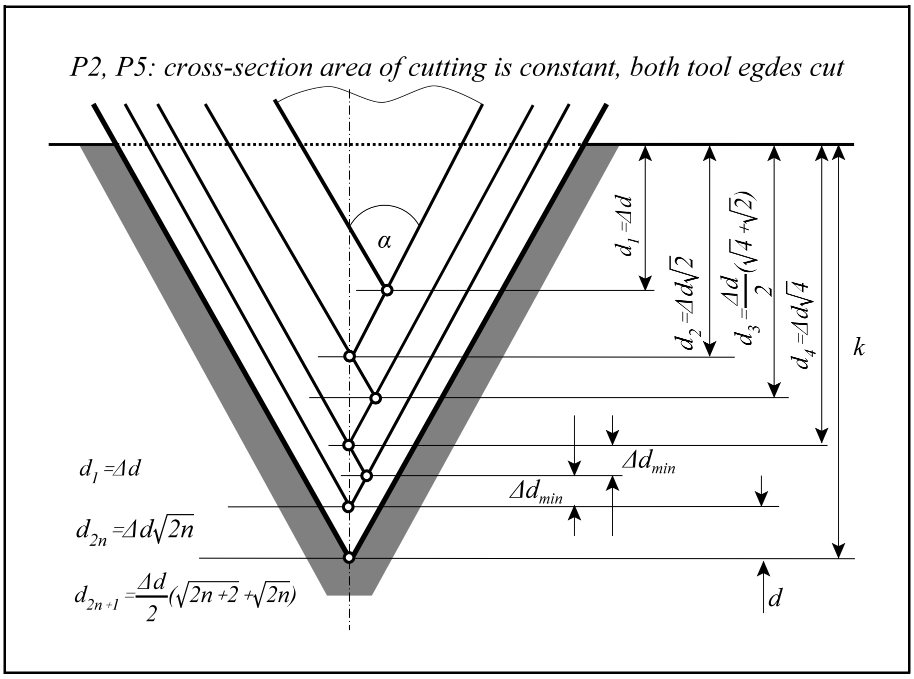
or

```
G76 X24 Z-50 K3 D1 I-4 A60 F3 P2 (method 2 of specification)
G0 X100 Z50
...
```



P1: cross-section area of cutting is constant, one of the tool egdes cuts
α
$d_1=\Delta d$
$d_n=\Delta d\sqrt{n}$
k
Δd_{min}
Δd_{min}
d

Fig. 18.2.7-3



P2, P5: cross-section area of cutting is constant, both tool egdes cut
α
$d_1=\Delta d$
$d_2=\Delta d\sqrt{2}$
$d_3=\frac{\Delta d}{2}(\sqrt{4}+\sqrt{2})$
$d_4=\Delta d\sqrt{4}$
k
Δd_{min}
Δd_{min}
d
$d_1=\Delta d$
$d_{2n}=\Delta d\sqrt{2n}$
$d_{2n+1}=\frac{\Delta d}{2}(\sqrt{2n+2}+\sqrt{2n})$

Fig. 18.2.7-4

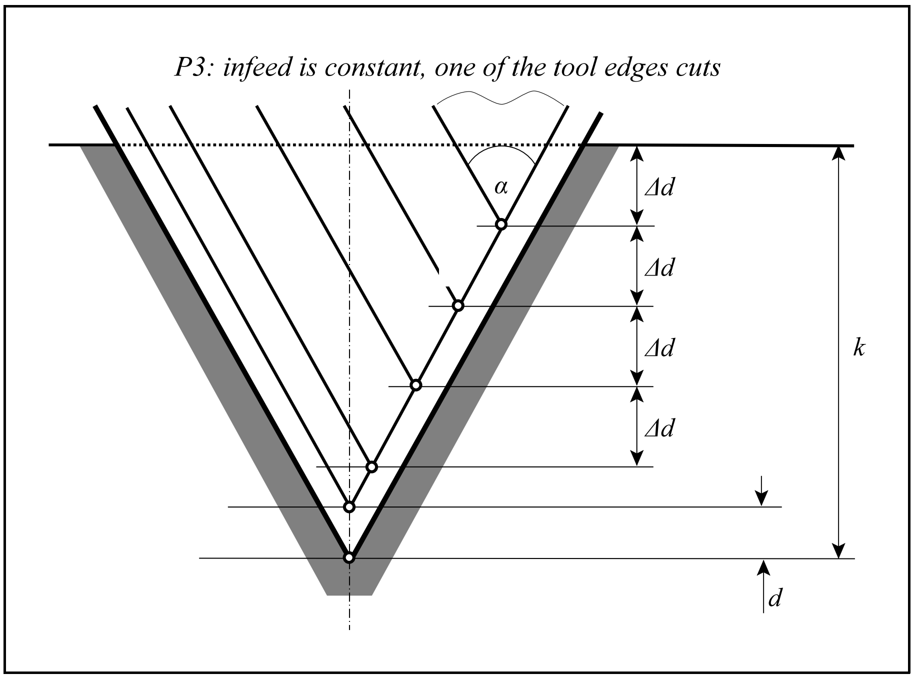


Fig. 18.2.7-5

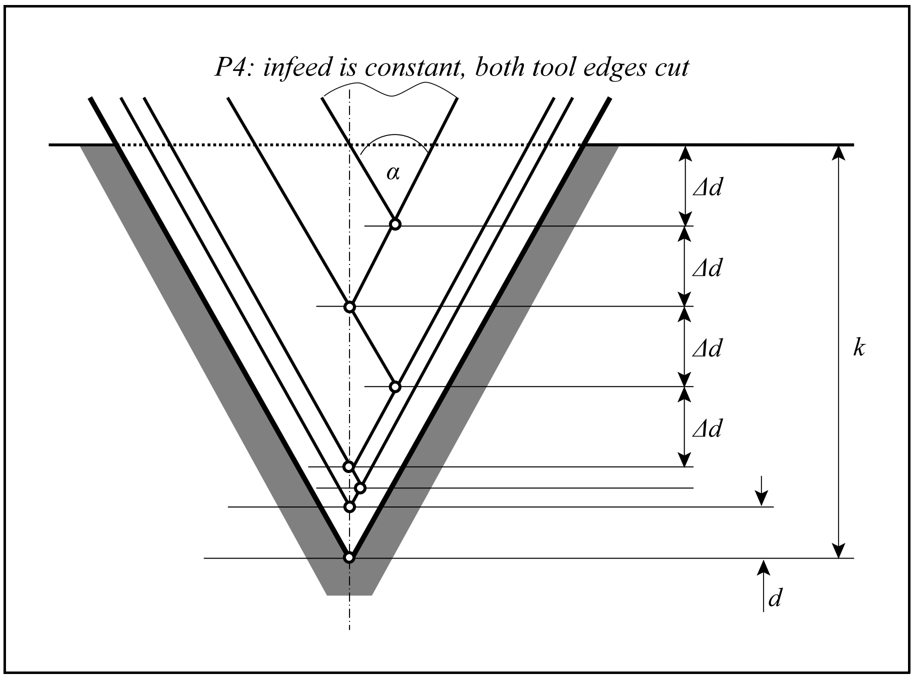


Fig. 18.2.7-6

19 Canned Cycles for Drilling

A canned cycle for drilling can consist of the following operations:

Operation 1: Positioning in the selected plane
Operation 2: Activity after positioning
Operation 3: Rapid traverse to the point R (point of approaching)
Operation 4: Activity at the point R
Operation 5: Drilling to the bottom point of the hole
Operation 6: Activity at the bottom point of the hole
Operation 7: Retracting to the point R
Operation 8: Activity at the point R
Operation 9: Rapid traverse retracting to the initial point
Operation 10: Activity at the initial point

Point R, point of approaching: The point to which the tool approaches the workpiece at rapid traverse rate.

Initial point: The position to which the drilling axis moves before starting the cycle.

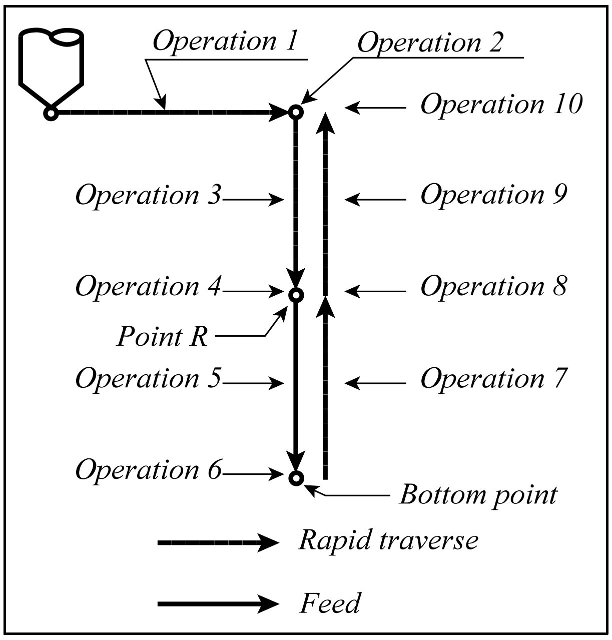


Fig. 19-1

The operations above describe drilling cycles generally; some of them can be omitted in specific cases.

The drilling cycles have ***positioning plane*** and ***drilling axis***. The drilling axis is designated by the plane selecting commands G17, G18 and G19. All the other axes are moved in the positioning plane.

Code G	Positioning plane	Drilling axis
G17	Plane X_p Y_p	Z_p
G18	Plane Z_p X_p	Y_p
G19	Plane Y_p Z_p	X_p

where: X_p: the axis X or an axis parallel with it
Y_p: the axis Y or an axis parallel with it
Z_p: the axis Z or an axis parallel with it
The axes U, V and W will be considered to be parallel axes if they are defined as parallel ones in the parameter N0103 Axis to Plane.
If face drilling is to be programmed where the drilling axis is the axis Z, the plane G17 will have to be selected; but if side drilling is to be programmed where the drilling axis is the axis X, the plane G19 will have to be selected.

Drilling cycles can be configured using commands **G98** and **G99**:
G98: in the course of the drilling cycle, ***the tool is retracted up to the initial point.*** It is a default position the control gets to after switching on, reset or deletion of the cycle mode.
G99: in the course of the drilling cycle, ***the tool is retracted up to the point R,*** therefore the operations 9 and 10 will be omitted.

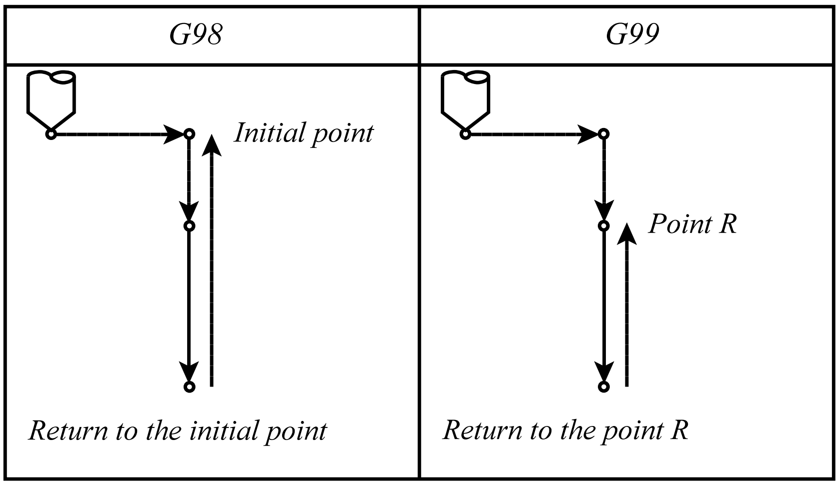


Fig. 19-2

The codes of the drilling cycles are: **G83.1, G84.1, G86.1, G81, ..., G89**
These codes set up the cycle mode enabling the cycle variables to be modal.
The code G80 cancels the cycle mode and deletes the stored cycle variables.

Addresses used in the drilling cycles and their meaning:

```
G17  G_   Xp_ Yp_ q_   I_ J_   Zp_ R_ Q_ E_ P_ F_ S_   L_
G18  G_   Zp_ Xp_ q_   K_ I_   Yp_ R_ Q_ E_ P_ F_ S_   L_
G19  G_   Yp_ Zp_ q_   J_ K_   Xp_ R_ Q_ E_ P_ F_ S_   L_
    ─┐   ────────┐    ────┐    ─────────────────────┐   │
     │           │        │                         │   └─ repetition number
     │           │        │                         └───── data of drilling
     │           │        └─────────────────── displacement after orientation
     │           └──────────────────────────── position of the hole
     └──────────────────────────────────────── code of drilling
```

The code of drilling:
Interpretation of the codes will be given later.
The codes will be modal until command G80 or a code belonging to the group 1 of the codes G (interpolation group: G01, G02, G03, G33) is programmed, or they will be deleted in case of mode change.
So long as the cycle state is on by the commands G83.1, G84.1, G86.1, G81, ..., G89, the modal cycle variable will also be modal between the drilling cycles of various type.

The initial point:
The initial point is the position of the axis selected for drilling, and it will be recorded:
– when the cycle mode is set up. For example, in the following case:

```
N1 G17 G90 G0 Z200
N2 G81 X0 C0 Z50 R150
N3 X100 C30 Z80
```

the position of the initial point will be Z=200 in either of the blocks N2 and N3.
– or, when a new drilling axis is selected. For example, in the following case:

```
N1 G17 G90 G0 Z200 W50
N2 G81 X0 C0 Z50 R150
N3 X100 C30 W20 R25
```

the position of the initial point is Z=200 in the block N2
the position of the initial point is W=50 in the block N3
In the case of changing the drilling axis, it is mandatory to program R otherwise the control will send the error message '2053 No bottom or R point'.

Position of the hole: X_p, Y_p, Z_p q
The entered coordinate values, excluding the drilling axis, will be considered by the control to be the position of the hole. They can be ***the main axes of selected plane***, ***the axes parallel with them***, or ***other axis not selected for drilling (q: for example C)***.
The entered values can be incremental or absolute ones given as orthogonal or polar coordinates with metric or inch dimension.
The mirroring, rotating and scaling commands are applicable to the entered coordinate values.
The control moves to the position of the hole at rapid traverse rate independently of which of the code of the group 1 is valid.

Displacement after orientation of the spindle: I, J, K

If the spindle can be oriented on a given machine, in the boring cycles G76 and G87 the tool can be retracted from the hole being shifted away from the surface in order not to scratch it. In this case, the direction of shifting can be specified at the addresses I, J and K. The addresses are interpreted by the control according to the selected plane:

G17:	I, J
G18:	K, I
G19:	J, K

The addresses are ***always*** interpreted as ***incremental orthogonal data***. The address can be given in metric unit or in inch.

The mirroring, rotating and scaling command are not applicable to the data I, J and K. The I, J and K are modal values. The code G80 or the codes of the interpolation group delete their values. The shifting is executed at rapid traverse rate.

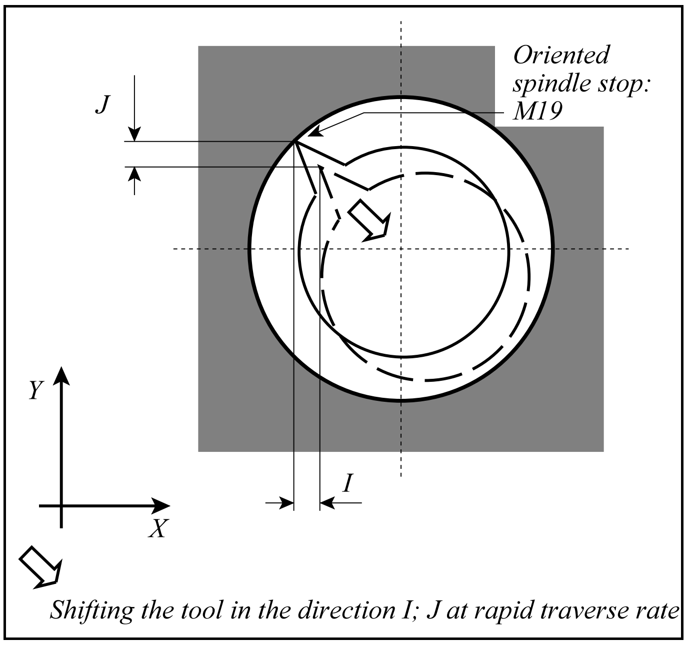

Oriented spindle stop: M19
J
I
Y
X
Shifting the tool in the direction I; J at rapid traverse rate

Fig. 19-3

Data of drilling:

Bottom point of the hole: X_p, Y_p, Z_p

The bottom point of the hole has to be specified at the address of the drilling axis. The coordinate of the bottom point of the hole is always interpreted as orthogonal data. It can be absolute or incremental given in metric unit or in inch. If the value of the bottom point is given as incremental one, the displacement will be calculated from the point R.

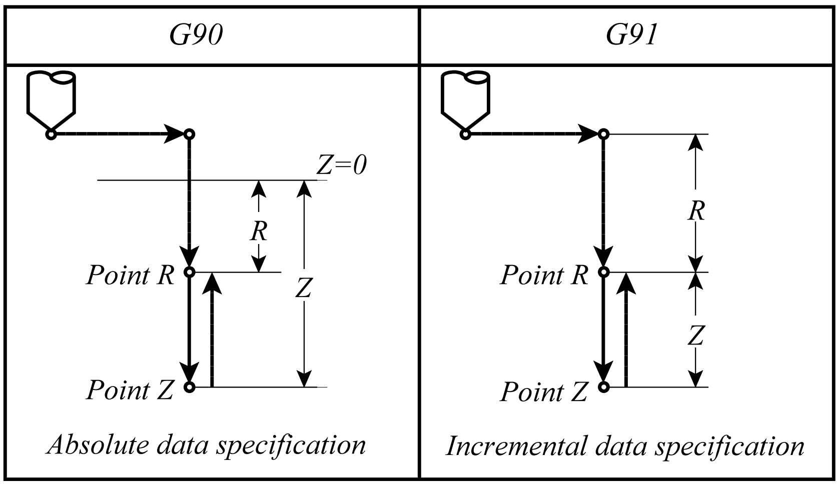


Fig. 19-4

The mirroring and scaling commands are applicable to the data of the bottom point. The data of the bottom point is a modal value. The code G80 or the codes of the interpolation group delete its value. The bottom point is always approached by the control at the actual feed rate being valid.

Point of approaching: R

The point of approaching has to be specified at the address R. The address R is always interpreted as orthogonal data. It can be absolute or incremental given in metric unit or in inch. If the address R is given as incremental data, its value will be calculated from the initial point. The mirroring and scaling data are applicable to the data of the point R. The data of the point R is a modal value. The code G80 or the codes of the interpolation group delete its value. The point R is always approached by the control at rapid traverse rate.

Depth of cut: Q

It is the value of the depth of cut in the cycles G83.1 and G83. It is always a positive incremental orthogonal data. The value of the depth of cut is a modal data. The code G80 or the codes of the interpolation group delete its value. The scaling command is not applicable to the depth of cut.

Auxiliary data: E

It is the extent of retraction in the cycle G83.1, and, in the cycle G83, it is the value motion to which executed at rapid traverse rate before infeed. It is always a positive incremental orthogonal data. The scaling command is not applicable to the auxiliary data. The value of the auxiliary data is a modal data. The code G80 or the codes of the interpolation group delete its value. If it not programmed, the control will have the required value from the parameters N1500 Return Val G73 and N1501 Clearance Val G83.

Dwell: P

It specifies the time of wait at the bottom of the hole. For specification of it, the rules described at G04 are valid. The value of dwell is a modal one. The code G80 or the codes of the interpolation group delete its value.

Feed: F

It specifies the feed. Its value is modal. Only programming another data F rewrites it. The code G80 or another code does not delete it.

Extraction override: I (%)

In the cycles G85, G89, G84.2, G84.3, extraction is generally executed at a feed rate programmed at the address F. At the address I, the extraction override can be specified as a percental value. If it is not programmed, the control will have the value of override from parameter.

Spindle speed: S

Its value is modal. Only programming another data S rewrites it. The code G80 or another code does not delete it.

Repetition number: L

It defines the number as many times the cycle is repeated throughout the range of 1–99999999. If L is not filled, the value L=1 will be taken into account by the control. In the case of L=0, the data of the cycle will be stored but will not be executed. The value of L is valid only in the block where it is specified.

Examples on modality of drilling codes and cycle variables:

```
        N1 G17 G0 Z_ M3
        N2 G81 X_ C_ Z_ R_ F_
```

At the beginning of the cycle mode, it is mandatory to specify the drilling data (Z, R).

```
        N3 X_
```

Since the drilling data were specified in the block N2 and the same ones are required in the block N3, it is not necessary to fill them, i.e. G81, Z_, R_, F_ can be omitted. The position of the hole changes in the direction X only, the drill bit moves in this direction, and then it drills such a hole just like one was drilled in the block N2.

```
        N4 G82 C_ Z_ P_
```

The position of the hole is shifted in the direction C. The method of drilling complies with the G82, the bottom point Z takes on a new value, the point of approaching and the feed (R and F) are taken from the block N2.

```
        N5 G80 M5
```

The cycle mode and the modal cycle variables will be deleted excluding F.

```
        N6 G85 C_ Z_ R_ P_ M3
```

Since the drilling data were deleted due to the command G80 in the block N5, the values Z, R and P have to be specified again.

```
        N7 G0 X_ C_
```

The cycle mode and the modal cycle variables will be deleted excluding F.

Examples on using the cycle repetition:

If holes of the same kind with equal spacing between them are to be drilled using the same parameters, the repetition number can be specified at the address L. The L is valid only in the block in which it is specified.

```
N1 G90 G19 G0 X300 Z40 C0 M3
N2 G91 G81 X-40 Z100 R-20 F50 L5
```

Due to the commands above, the control drills 5 holes of the same kind along the axis Z with spacing of 100 mm between them. The position of the first hole is Z=140 and C=0. Since this is side drilling (drilling in the direction of the axis X), the plane G19 was selected.

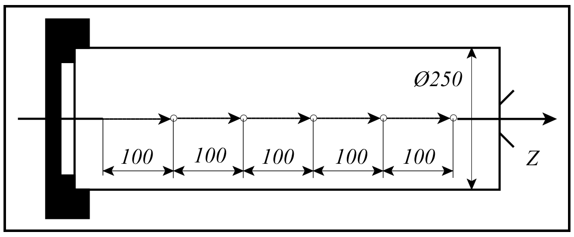


Fig. 19-5

Due to the G91, the position of the hole is given incrementally.

```
N1 G90 G17 G0 X200 C-60 Z50
N2 G81 CI60 Z-40 R3 F50 L6
```

In accordance with the commands above, the control drills 6 holes along the bolt hole with the diameter of 100 mm, with spacing of 60° between them. The position of the first hole is X=200 and C=0. Since this is face drilling (drilling in the direction of the axis Z), the plane G17 was selected.

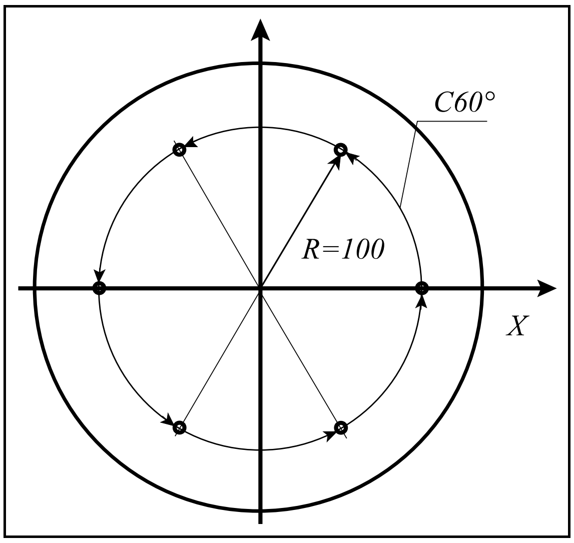


Fig. 19-6

19.1 Detailed Description of the Drilling Cycles

19.1.1 High-speed Peck Drilling Cycle (G83.1)

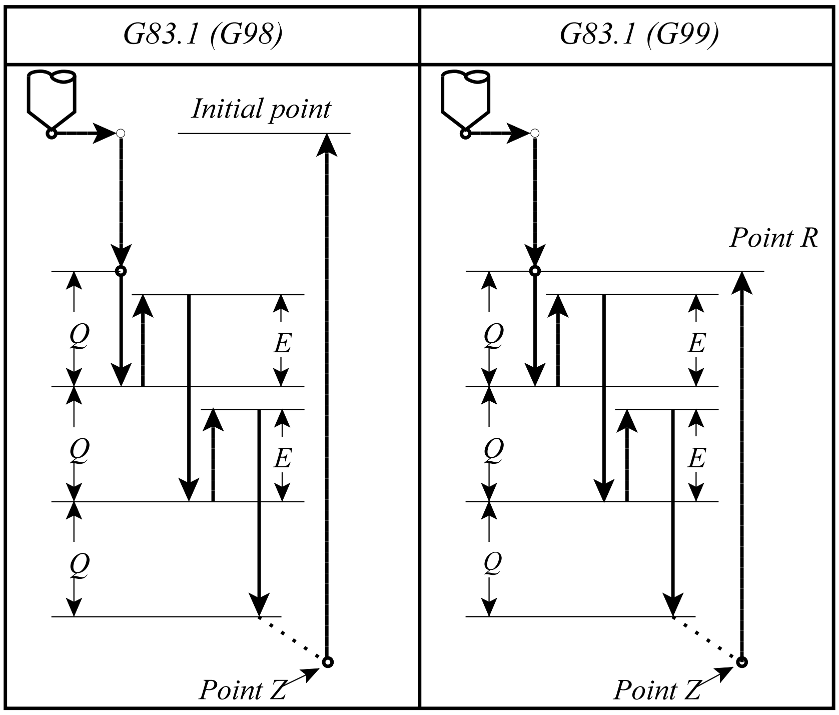


Fig. 19.1.1-1

The variables used in the cycle are the following:

G17 **G83.1** X_p__ Y_p__ Z_p__ R__ Q__ E__ F__ L__
G18 **G83.1** Z_p__ X_p__ Y_p__ R__ Q__ E__ F__ L__
G19 **G83.1** Y_p__ Z_p__ X_p__ R__ Q__ E__ F__ L__

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F
Operation 6:	–
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	–
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

Execution of the Operation 5 of drilling is as follows:

– drilling the ***depth of cut*** specified at the address ***Q*** into the workpiece at the feed rate;
– ***retracting*** with a value specified at the address ***E*** or in the parameter N1500 Return Val G73, at rapid traverse rate;
– drilling the depth of cut Q again into the workpiece from the bottom point of the previous drilling;
– retracting with a value specified at the address E at rapid traverse rate.

The drilling proceeds up to the bottom point specified at the address Z.

19.1.2 Left-Handed Tapping Cycle Using Spring Tap (G84.1)

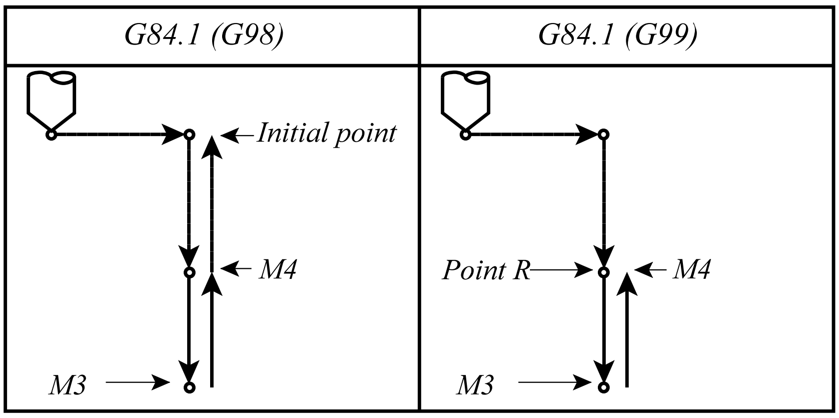


Fig. 19.1.2-1

Using this cycle is allowed when spring tap is applied.

The variables used in the cycle are the following:

G17 **G84.1** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G84.1** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G84.1** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

Prior to starting the cycle, the direction of spindle rotation M4 (counter-clockwise) must be switched on and programmed.
The value of feed has to be specified according to the lead of the tap:
– in the state G94 (feed per minute):

$$F=P\times S$$

where:	P: the lead of the thread in mm/rev or inch/rev;
	S: the spindle speed in rev/min.

– in the state G95 (feed per revolution):

$$F=P$$

where:	P: the lead of the thread in mm/rev or inch/rev.

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F, ***override and stop are disabled***
Operation 6:	Dwell for the value specified at the address P Reversal of spindle rotation: M3
Operation 7:	Retracting to the point R at feed rate F, ***override and stop are disabled***
Operation 8:	Reversal of spindle rotation: M4
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

19.1.3 Boring Cycle with Automatic Tool Shift (G86.1)

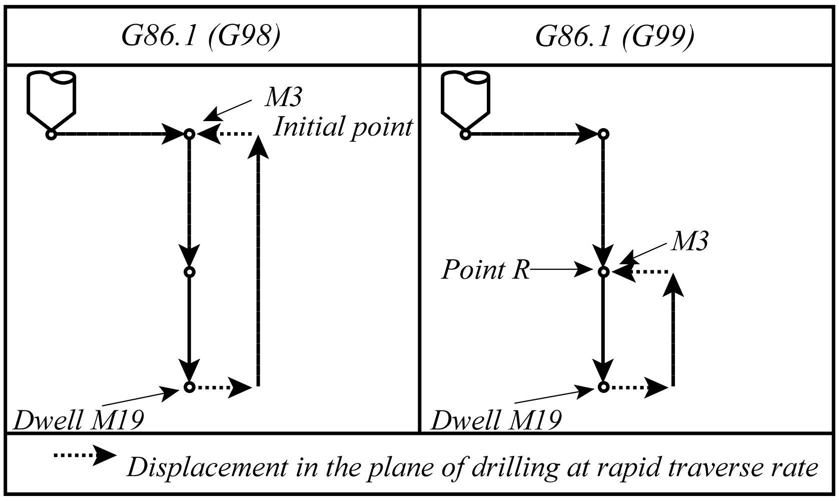


Fig. 19.1.3-1

The cycle G86.1 can only be used if spindle orientation is built in the machine tool. It is the state 1 of the bit ORI of the parameter N0607 Spindle Config which indicates this capability to the control. Otherwise, the control will send the error message '2137 G76, G87 error'.
Since spindle orientation and tool shift specified at the addresses I, J and K are executed by the cycle after the boring, the surface will not be scratched during tool retraction.
The variables used in the cycle are the following:

G17 **G86.1** X_p__ Y_p__ I__ J__ Z_p__ R__ P__ F__ L__
G18 **G86.1** Z_p__ X_p__ K__ I__ Y_p__ R__ P__ F__ L__
G19 **G86.1** Y_p__ Z_p__ J__ K__ X_p__ R__ P__ F__ L__

Prior to starting the cycle, the command M3 has to be issued.
The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Dwell for the value specified at the address P Spindle orientation: M19 In the selected plane, tool shift with the values specified at the addresses I, J and K, at rapid traverse rate
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	In the case of G99: In the selected plane, tool reset with the values specified at the addresses I, J and K, at rapid traverse rate Restarting the spindle in the direction M3
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	In the case of G98: In the selected plane, tool reset with the values specified at the addresses I, J and K, at rapid traverse rate Restarting the spindle in the direction M3

19.1.4 Cancelling the Cycle State (G80)

Due to this code, the cycle state will be cancelled, and the cycle variables will be deleted.
The Z and the R take on incremental 0, all the rest variables take on 0.
If coordinates are programmed in the block **G80** and any other command is nott issued, the motion will be executed according to the interpolation code (the group 1 of the codes G or the interpolation group) that was valid prior to activating the cycle.

19.1.5 Drilling Cycle with Retraction at Rapid Traverse Rate (G81)

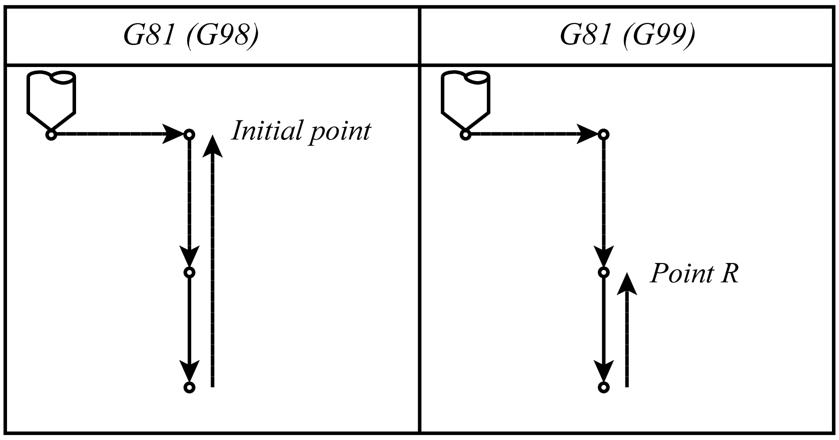


Fig. 19.1.5-1

The variables used in the cycle are the following:

G17 **G81** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G81** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G81** Y_p__ Z_p__ X_p__ R__ F__ L__

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F
Operation 6:	–
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	–
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

19.1.6 Drilling Cycle with Dwell and with Retraction at Rapid Traverse (G82)

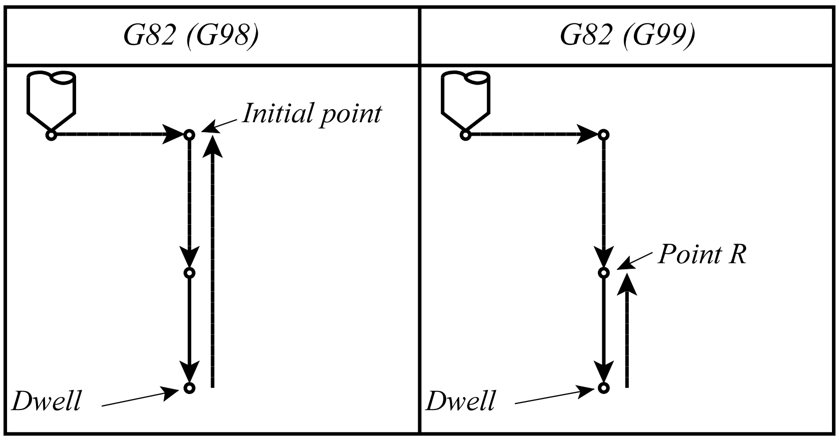


Fig. 19.1.6-1

The variables used in the cycle are the following:

G17 **G82** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G82** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G82** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F
Operation 6:	Dwell for the value specified at the address P
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	–
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

19.1.7 Peck Drilling Cycle (G83)

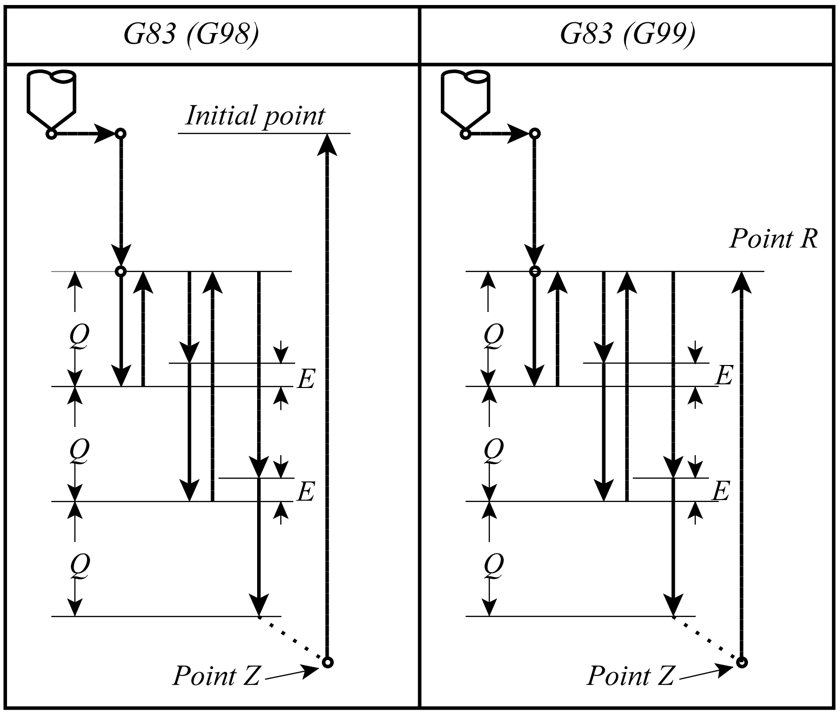


Fig. 19.1.7-1

The variables used in the cycle are the following:

G17 **G83** X_p__ Y_p__ Z_p__ R__ Q__ E__ F__ L__
G18 **G83** Z_p__ X_p__ Y_p__ R__ Q__ E__ F__ L__
G19 **G83** Y_p__ Z_p__ X_p__ R__ Q__ E__ F__ L__

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F
Operation 6:	–
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	–
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

Execution of the Operation 5 of drilling is as follows:

– drilling the ***depth of cut*** specified at the address ***Q*** into the workpiece at the feed rate;
– retracting to the point R, at rapid traverse rate;
– ***approaching*** the previous depth ***to a distance E***;
– drilling the depth Q again into the workpiece from the bottom point of the previous drilling, at the feed rate (displacement is E+Q);
– retracting to the point R, at rapid traverse rate.

The drilling proceeds up to the bottom point specified at the address Z.
The distance E is taken from the address ***E*** or from the parameter N1501 Clearance Val G83.

19.1.8 Right-Handed Tapping Cycle Using Spring Tap (G84)

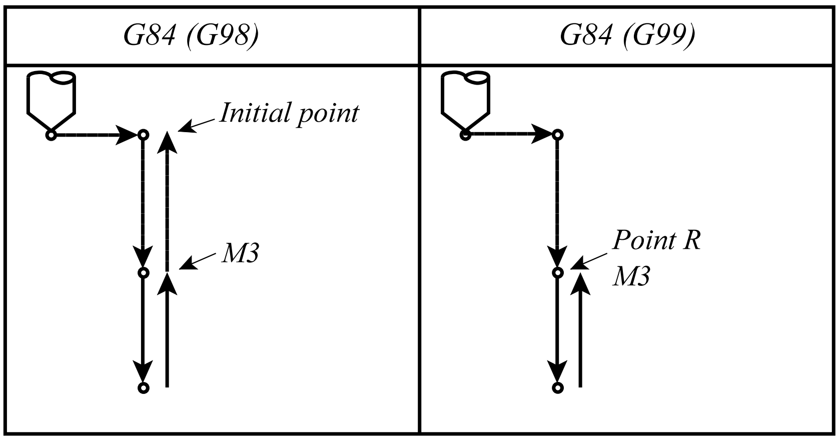


Fig. 19.1.8-1

Using this cycle is allowed when spring tap is applied.

The variables used in the cycle are the following:

G17 **G84** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G84** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G84** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

Prior to starting the cycle, the direction of spindle rotation M3 (clockwise) must be switched on.
The value of feed has to be specified according to the lead of the tap:
– in the state G94 (feed per minute):

$$F=P\times S$$

where: P: the lead of the thread in mm/rev or inch/rev;
S: the spindle speed in rev/min.

– in the state G95 (feed per revolution):

$$F=P$$

where: P: the lead of the thread in mm/rev or inch/rev.

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Drilling to the bottom point at the feed rate F, ***override and stop are disabled***
Operation 6:	Dwell for the value specified at the address P Reversal of spindle rotation: M4
Operation 7:	Retracting to the point R at feed rate F, ***override and stop are disabled***
Operation 8:	Reversal of spindle rotation: M3
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

19.1.9 Rigid Tapping Cycle (G84.2, G84.3)

Cycles of rigid tapping thread can be applied only on the machines the spindle of which is equipped with an encoder for positioning, and the spindle drive can be fed back for position control (the value of the bit INX of the parameter N0607 Spindle Config is 1). Otherwise, the control will send the error message '2138 Spindle can not be indexed'.

In the case of rigid tapping, the quotient of the feed of the drill axis and the spindle speed must be equal to the thread lead of the tap. In other words, under ideal conditions of rigid tapping, the quotient F=P/S must be constant from moment to moment,

where: P: the lead of the thread in mm/rev or inch/rev;
F: the feed in mm/min or inch/min;
S: the spindle speed in rev/min.

In the cycles G84.1 and G84 (cycle of tapping left-hand thread using spring tap and cycle of tapping left-hand thread using spring tap, respectively), the spindle speed and the feed of the drill axis are controlled independently of each other. Accordingly, the requirement mentioned above cannot be met to the full. It is particularly true at the bottom of the hole where the feed of the drill axis and the spindle speed should be decreased up to zero and then, in the opposite direction, they should be increased, in synchronism with each other. In the case above, as far as the control technique is concerned, complying with this condition is not possible at all. This problem can be avoided by inserting the tap into a spring balancing adapter to be set in the spindle; this adapter compensates fluctuation of the value of the quotient F/S.

The principle of control is different in the case of the cycles G84.2 and G84.3 that make it possible to avoid the use of spring tap. In these cycles, the control continuously maintains the constant value of the quotient F/S from moment to moment.
As far as the control technique is concerned, in the former case the control regulates only the spindle speed, while in the latter case, it regulates the spindle position too. In the cycles G84.2 and G84.3, the motions of the drill axis and the spindle are connected with each other using linear interpolation. With this method, the constant value of the quotient F/S is maintained in the phases of acceleration and deceleration too.

G84.2: right-handed rigid tapping cycle
G84.3: left-handed rigid tapping cycle

The variables used in the cycle are the following:

G17 **G84._** X_p__ Y_p__ Z_p__ R__ P__ F__ I__ S__ L__
G18 **G84._** Z_p__ X_p__ Y_p__ R__ P__ F__ I__ S__ L__
G19 **G84._** Y_p__ Z_p__ X_p__ R__ P__ F__ I__ S__ L__

At the end of the cycle, the spindle remains in the indexed state (the position control loop is closed); it is the programmer who has to restart it, if necessary.

The value of feed has to be specified according to the lead of the tap:
– in the state G94 (feed per minute):

$$F=P\times S$$

where: P: the lead of the thread in mm/rev or inch/rev;
S: the spindle speed in rev/min.

– in the state G95 (feed per revolution):

F=P

where: P: the lead of the thread in mm/rev or inch/rev.

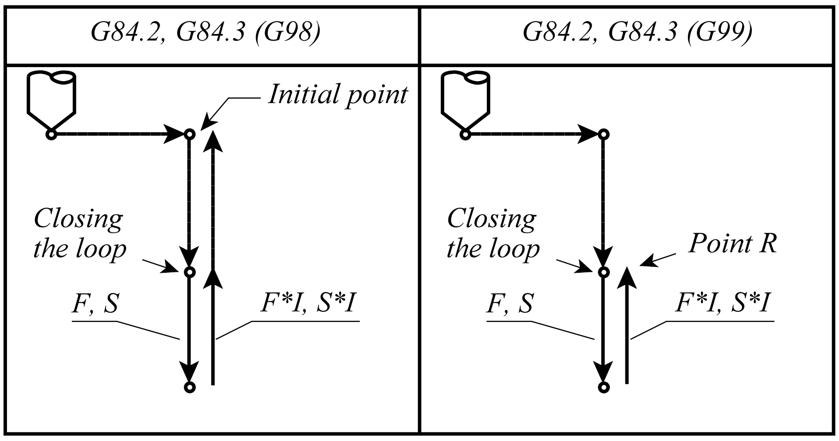


Fig. 19.1.9-1

The operations of the cycle in the case of G84.2 are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	If the value of the bit TSC of the parameter N1503 Drilling Cycles Config. =0, the control will not orientate the spindle and will only close the position control loop. The code of closing the loop is determined by the parameter N0823 M Code for Closing S Loop. This code will be transmitted by the control to the PLC (faster execution). =1, the control will orientate the spindle prior to tapping and will transmit the command M19 to the PLC (finding the way back into the thread).
Operation 5:	Linear interpolation between the drill axis and the spindle in the clockwise direction of rotation (+) in the case of G84.2, and in the counter-clockwise direction of rotation (–) in the case of G84.3.
Operation 6:	Dwell for the value specified at the address P
Operation 7:	Linear interpolation between the drill axis and the spindle in the counter-clockwise direction of rotation (–) in the case of G84.2, and in the clockwise direction of rotation (+) in the case of G84.3.

Extraction override I (%)

The extraction feed override being either programmed at the address I or taken from parameter will be effective in the only case if it is enabled by writing the value 1 at the bit #4 EOE of the parameter N1503 Drilling Cycles Config.

If value is not given at the address I in the block, the extraction feed override in the operation 7 will be the value of the parameter N1506 Extraction Override in Tapping given in %.The feed override button also produces effect on speed calculated in such a way, so the value of feed will be calculated according to the following formula:

Fextraction=Fprogrammed×Feed override×Extraction Override in Tapping/100

Acceleration during extraction
During extraction, a value different from (smaller than) the acceleration value set for the spindle can be specified in the parameter N1507+n Rn Acc in Tapping if it is enabled by writing the value 1 at the bit #6 EAE of the parameter N1503 Drilling Cycles Config.. In this case, the acceleration value will be taken from the parameters N1523+n Rn Acc in Extract being different range by range (n=1 ... 8).

Operation 8:	–
Operation 9:	In the case of G98, retracting to the initial point at rapid traverse rate
Operation 10:	–

19.1.10 Peck Rigid Tapping Cycle (G84.2, G84.3)

The code of the cycle is as follows:

G84.2 Q__: right-handed peck rigid tapping cycle
G84.3 Q__: left-handed peck rigid tapping cycle

In the cycle, the control will apply ***chip breaking in the case*** if ***depth of cut*** is programmed ***at the address Q***.

The variables used in the cycle are the following:

G17 **G84._** X_p__ Y_p__ Z_p__ R__ Q__ E__ P__ F__ S__ I__ L__
G18 **G84._** Z_p__ X_p__ Y_p__ R__ Q__ E__ P__ F__ S__ I__ L__
G19 **G84._** Y_p__ Z_p__ X_p__ R__ Q__ E__ P__ F__ S__ I__ L__

At the end of the cycle, the spindle remains in the indexed state (the position control loop is closed); it is the programmer who has to restart it, if necessary. The meaning of the codes **G98** and **G99** is the same as it was in the case of all the other drilling cycles.

It is the bit #3 PTC of the parameter N1503 Drilling Cycles Config. that determines the ***manner of chip breaking***. If the value of the bit #3 PTC is:

=0, fast chip breaking (according to the G83.1) will be applied;
=1, normal chip breaking (according to the G83) will be applied, and extraction will be executed to the point R.

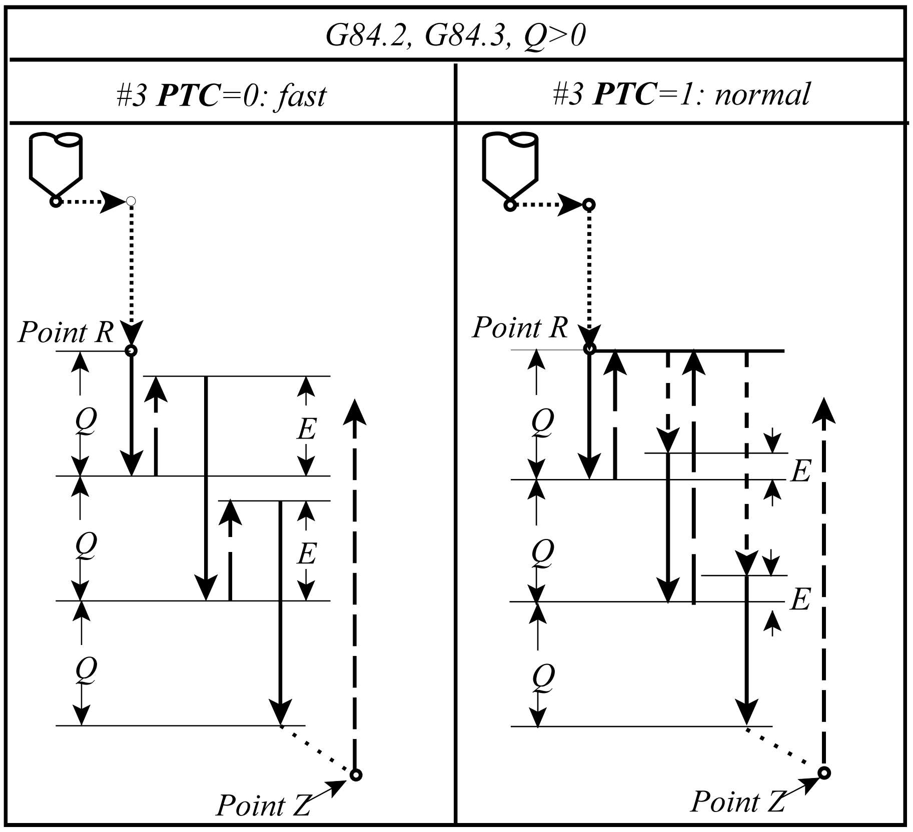


Fig. 19.1.10-1

Interpretation of the address E

In the case of the ***fast chip breaking*** (PTC=0), it is the address E where the ***value of the returning the tool*** can be specified. If the address E is not filled, the control will have the distance from the parameter N1504 Return Val in Tapping.

In the case of the ***normal chip breaking*** (PTC=1), it is the address E where the ***value of the approaching distance*** after return can be specified. If the address E is not filled, the control will have the distance from the parameter N1505 Clearance Val in Tapping.

Extraction override I (%)

In all the extraction phases, the extraction feed override, being either programmed at the address I or taken from parameter, will be effective in the only case if it is enabled by writing the value 1 at the bit #4 EOE of the parameter N1503 Drilling Cycles Config.

If value is not given at the address I in the block, the extraction feed override will be the value of the parameter N1506 Extraction Override in Tapping given in %.The feed override button also produces effect on speed calculated in such a way, so the value of feed will be calculated according to the following formula:

Fextraction=Fprogrammed×Feed override×Extraction Override in Tapping/100

Acceleration during extraction

During extraction, a value different from (smaller than) the acceleration value set for the spindle can be specified in the parameter N1507+n Rn Acc in Tapping if it is enabled by writing the value 1 at the bit #6 EAE of the parameter N1503 Drilling Cycles Config.. In this case, the acceleration value will be taken from the parameters N1523+n Rn Acc in Extract being different range by range (n=1 ... 8).

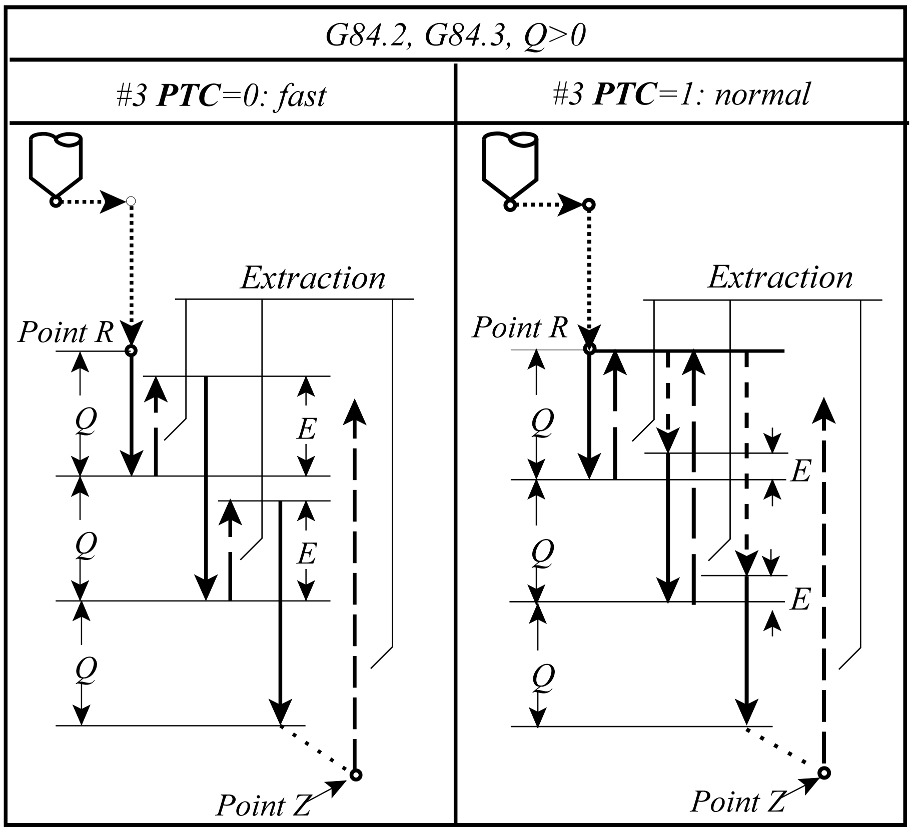


Fig. 19.1.10-2

Return override (%) in the case of normal chip breaking

In the case of normal chip breaking (the bit #3 PTC of the parameter N1503 Drilling Cycles Config. is 1), in the course of ***returning,*** application of an override different from 100% for the programmed F will be allowed if it is enabled by writing the value 1 at the bit #5 ROE of the parameter N1503 Drilling Cycles Config..
Thus, the control will have the value of override from the value in % of the parameter N1507 Return Override in Tapping.

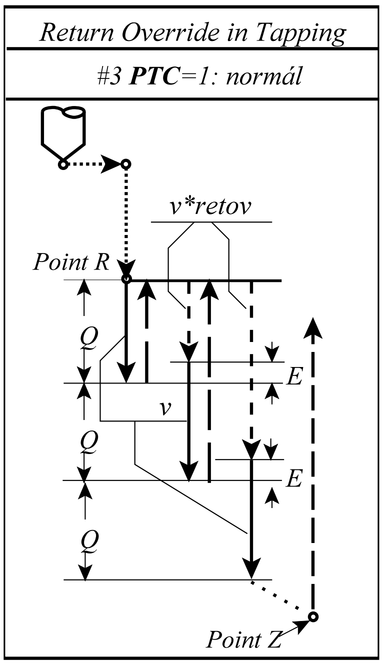


Fig. 19.1.10-3

The subject-matter mentioned in the previous subchapter apply to the other input parameters of the cycle.

19.1.11 Boring Cycle with Retraction at Feed Rate (G85)

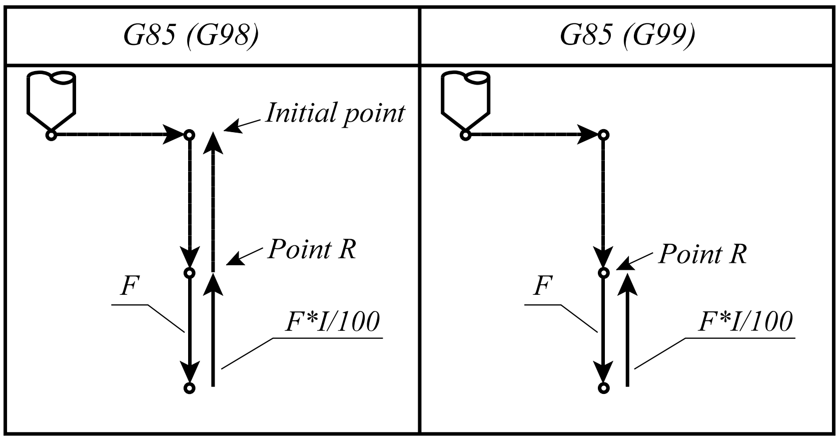


Fig. 19.1.11-1

The variables used in the cycle are the following:

G17 **G85** X_p__ Y_p__ Z_p__ R__ F__ I__ L__
G18 **G85** Z_p__ X_p__ Y_p__ R__ F__ I__ L__
G19 **G85** Y_p__ Z_p__ X_p__ R__ F__ I__ L__

The operations of the cycle are the following:

Operation 1: Positioning in the selected plane at rapid traverse rate
Operation 2: –
Operation 3: Rapid traverse to the point R (point of approaching)
Operation 4: –
Operation 5: Boring to the bottom point at the feed rate F
Operation 6: –
Operation 7: Retracting to the point R at the feed rate F×I/100
Extraction override I (%)
If value is not given at the address I in the block, the extraction feed override in the operation 7 will be the value of the parameter N1502 Extraction Override in G85, G89 .The feed override button also produces effect on speed calculated in such a way, so the value of feed will be calculated according to the following formula:

Fextraction=Fprogrammed×Feed override×Extraction Override in G85, G89/100

Operation 8: –
Operation 9: In the case of G98, retracting to the initial point at rapid traverse rate
Operation 10: –

19.1.12 Boring Cycle with Retraction with Spindle in Standstill (G86)

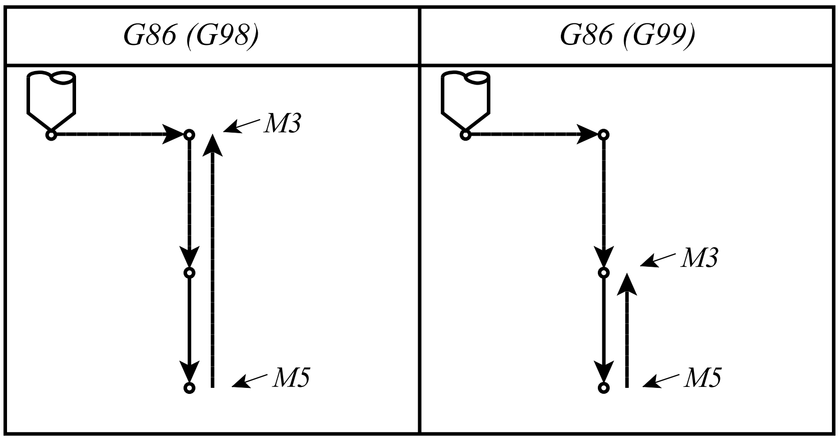


Fig. 19.1.12-1

The variables used in the cycle are the following:

G17 **G86** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G86** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G86** Y_p__ Z_p__ X_p__ R__ F__ L__

During starting the cycle, the direction of rotation M3 has to be specified for the spindle.
The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Stopping the spindle: M5
Operation 7:	In the case of G99: Retracting to the point R at rapid traverse rate
Operation 8:	In the case of G99: Restarting the spindle in the direction M3
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	In the case of G98: Restarting the spindle in the direction M3

19.1.13 Manual Control/Back Boring Cycle (G87)

The cycle is executed by the control in two different ways.

A. Drilling cycle with manual control at the bottom point

In the case, if ***spindle orientation is not built in the machine tool***, i.e. the bit #1 ORI of the parameter N0607 Spindle Config is 0, the the control will operate according to the case A.

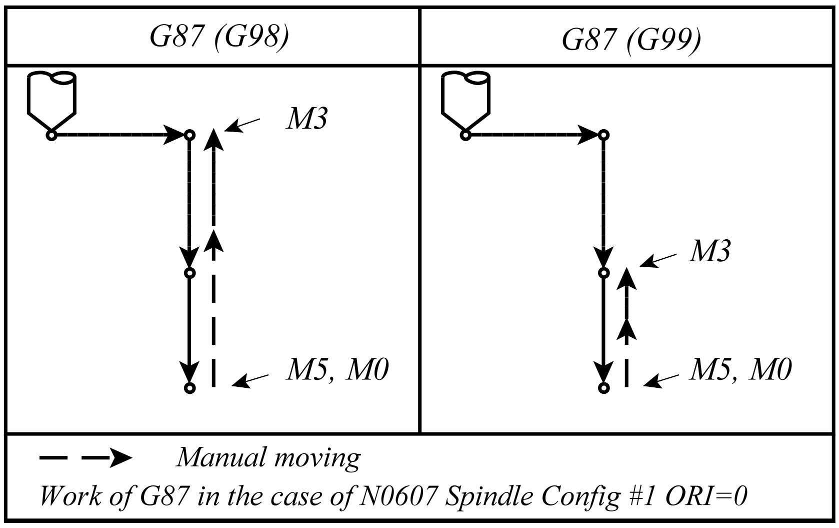


Fig. 19.1.13-1

The variables used in the cycle are the following:

G17 **G87** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G87** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G87** Y_p__ Z_p__ X_p__ R__ F__ L__

During starting the cycle, the direction of rotation M3 has to be specified for the spindle.
The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Stopping the spindle: M5 The control gets to the state STOP (M0), from where, having transferred to one of the manual modes (JOG, INCREMENTAL JOG, HANDWHEEL), the operator can operate the machine manually, i.e. he can shift the tool tip from the surface of the hole and extract the tool from the hole. Then, having returned to the AUTO mode, machining can be continued by start.
Operation 7:	In the case of G99: After START, retracting to the point R at rapid traverse rate
Operation 8:	In the case of G99: Restarting the spindle in the direction M3
Operation 9:	In the case of G98: After START, retracting to the initial point at rapid traverse rate
Operation 10:	In the case of G98: Restarting the spindle in the direction M3

B. Back boring with automatic shifting the tool

In the case, if ***spindle orientation is built in the machine tool***, i.e. the bit #1 ORI of the parameter N0607 Spindle Config is 1, the the control will operate according to the case B.

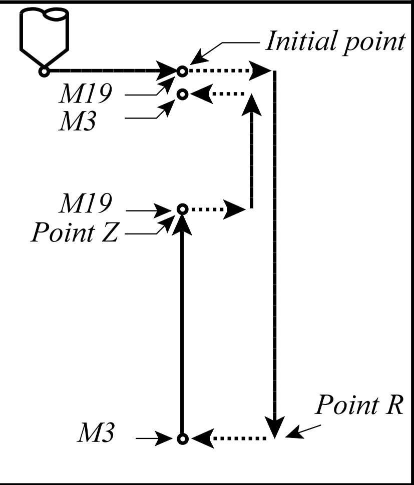
G87 (G98)	*G87 (G99)*
 Initial point M19 M3 M19 Point Z Point R M3	*Not used*
⋯➤ *Displacement in the plane of drilling at rapid traverse rate*	
Work of G87 in the case of N0607 Spindle Config #1 ORI=0	

Fig. 19.1.13-2

The variables used in the cycle are the following:

G17 **G87** X_p__ Y_p__ I__ J__ Z_p__ R__ F__ L__
G18 **G87** Z_p__ X_p__ K__ I__ Y_p__ R__ F__ L__
G19 **G87** Y_p__ Z_p__ J__ K__ X_p__ R__ F__ L__

During starting the cycle, the direction of rotation M3 has to be specified for the spindle.
The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	Orientating the spindle. In the selected plane, tool shift with the values specified at the addresses I, J and K, at rapid traverse rate
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	In the selected plane, tool reset with the values specified at the addresses I, J and K, at rapid traverse rate. Restarting the spindle in the direction M3
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Orientating the spindle: M19. In the selected plane, tool shift with the values specified at the addresses I, J and K, at rapid traverse rate
Operation 7:	–
Operation 8:	–
Operation 9:	Retracting to the initial point at rapid traverse rate
Operation 10:	In the selected plane, tool reset with the values specified at the addresses I, J and K, at rapid traverse rate. Restarting the spindle in the direction M3

It follows from the nature of the cycle, that, in contrast with the foregoing, ***the point of approaching, i.e. the point R is located lower than the bottom point***. This has to be taken into account in programming the addresses of the drill axis and R.
Since spindle orientation and tool shift specified at the addresses I, J and K are executed by the cycle prior to the boring, the tool break can be avoided during entering the hole.

19.1.14 Boring Cycle with Dwell and Manual Operation at the Bottom (G88)

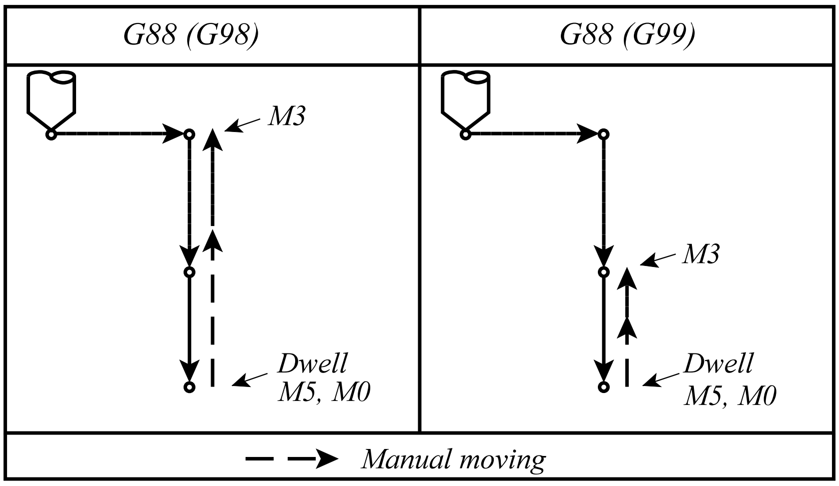


Fig. 19.1.14-1

The variables used in the cycle are the following:

G17 **G88** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G88** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G88** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

During starting the cycle, the direction of rotation M3 has to be specified for the spindle.
The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Dwell for the value specified at the address P Stopping the spindle: M5 The control gets to the state STOP (M0), from where, having transferred to one of the manual modes (JOG, INCREMENTAL JOG, HANDWHEEL), the operator can operate the machine manually, i.e. he can shift the tool tip from the surface of the hole and extract the tool from the hole. Then, having returned to the AUTO mode, machining can be continued by start.
Operation 7:	In the case of G99: After START, retracting to the point R at rapid traverse rate
Operation 8:	In the case of G99: Restarting the spindle in the direction M3
Operation 9:	In the case of G98: After START, retracting to the initial point at rapid traverse rate
Operation 10:	In the case of G98: Restarting the spindle in the direction M3

The cycle is the same as the case A of the G87, but there is dwell prior to stopping the spindle.

19.1.15 Boring Cycle with Dwell and with Retraction at Feed Rate (G89)

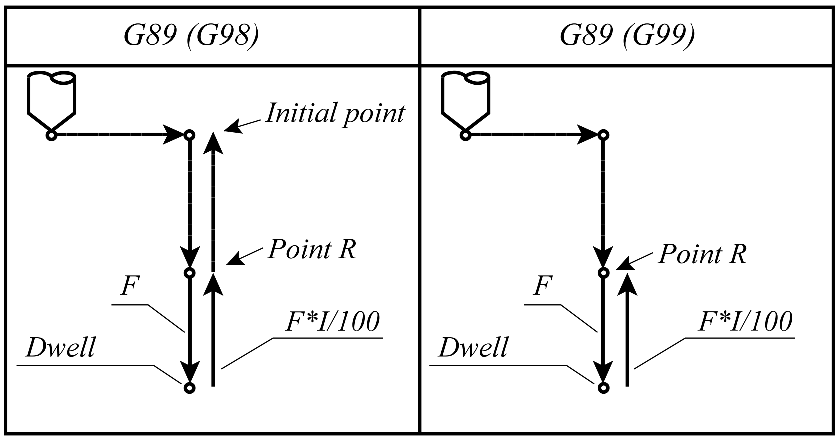


Fig. 19.1.15-1

The variables used in the cycle are the following:

G17 **G89** X_p__ Y_p__ Z_p__ R__ P__ F__ I__ L__
G18 **G89** Z_p__ X_p__ Y_p__ R__ P__ F__ I__ L__
G19 **G89** Y_p__ Z_p__ X_p__ R__ P__ F__ I__ L__

The operations of the cycle are the following:

Operation 1:	Positioning in the selected plane at rapid traverse rate
Operation 2:	–
Operation 3:	Rapid traverse to the point R (point of approaching)
Operation 4:	–
Operation 5:	Boring to the bottom point at the feed rate F
Operation 6:	Dwell for the value specified at the address P
Operation 7:	Retracting to the point R at the feed rate F×I/100 Extraction override I (%) If value is not given at the address I in the block, the extraction feed override in the operation 7 will be the value of the parameter N1502 Extraction Override in G85, G89 .The feed override button also produces effect on speed calculated in such a way, so the value of feed will be calculated according to the following formula:

Fextraction=Fprogrammed×Feed override×Extraction Override in G85, G89/100

Operation 8:	–
Operation 9:	In the case of G98: Retracting to the initial point at rapid traverse rate
Operation 10:	–

The cycle is equal to the G85, excluding dwell.

19.2 Remarks on the Use of the Drilling Cycles

– In cycle mode, the drilling cycle will be executed if a block without code G contains any of the following addresses:

X_p, Y_p, Z_p or q

where q is an optional axis, but not a drill axis.
Otherwise, the drilling cycle will not be executed.

– In cycle mode, if dwell block G04 P is programmed, the command will be executed according to the P, but the cycle variable related to dwell ***will not be deleted and rewritten***.

– The value of I, J, K, Q, E, Phas to be given in the blocks where drilling also occurs, otherwise the values ***will not be stored***.

An example to illustrate the foregoing:

```
G81 X_ C_ Z_ R_ F  (the drilling cycle will be executed)
    X              (the drilling cycle will be executed)
    F_             (the  drilling  cycle  will  not  be
                   executed, the F will be rewritten)
    M S_           (the  drilling  cycle  will  not be
                   executed, the code M will be executed)
 G4      P_        (the  drilling  cycle  will not be
                   executed, the dwell will be executed,
                   the dwell variable of the cycle will
                   not be rewritten)
    I Q            (the  drilling  cycle  will  not be
                   executed, the programmed values will
                   not be recorded as cycle variables)
```

– If a function is also programmed together with the drilling cycle, the function will be executed at the end of the first operation, after completion of positioning. If L is also programmed in the cycle, the function will be executed only in the first pass.

– The repetition number L is not modal.

– In the block by block mode, the control stops after the operations 1, 3 and 10 within the cycle.

– The button STOP does not produces effect in the operations 5, 6 and 7 of the cycles G74 and G84. If the button STOP is pushed during these operations, the control will continue its functioning, and it can be stopped at the end of the operation 7.

– In the operations 5, 6 and 7 of the cycles G84.1 and G84, the feed override and the spindle override is always 100% independently of the position of the switch.

– If Tnnmm is programmed in the cycle block, the new length compensation will be taken into account either during planar positioning or during drilling.

20 Functions to Control Axes

20.1 Polygonal Turning

In the case of polygonal turning, both the tool and the workpiece are rotated in relation to each other by a specified revolution ratio. Changing the revolution ratio and the number of cutting edges of the rotating tool, polygons with different number of sides can be produced. The product of the revolution ratio and the number of sides gives the number of sides of the polygon generated. For example, if the ratio between revolutions of the tool and the workpiece is 2:1 and the number of cutting edges is 3, a hexagon will be turned. Hexagon bolt or nut can simply be produced in this way by turning, for example. This method of machining is much faster than milling the polygon using polar coordinate interpolation, but the resulting side surfaces are not exact planes.

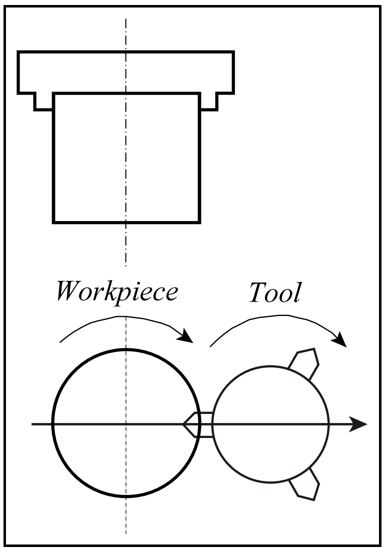


Fig. 20.1-1

20.1.1 Principle of Poligonal Turning

Let the rotational axis of the workpiece be on the coordinates X=0, Y=0.
Let the rotational axis of the tool be the point P_0 with the coordinates X=A, Y=0 which means that the distance between the rotational axis of the tool and the rotational axis of the workpiece is A.
Let the radius of the rotating tool be B. The nose of the tool is marked by P_t. In the moment t=0, the coordinates of the nose of the tool are X=A–B, Y=0.
Let the angular velocities of the workpiece rotation and the tool rotation be α and β, respectively.

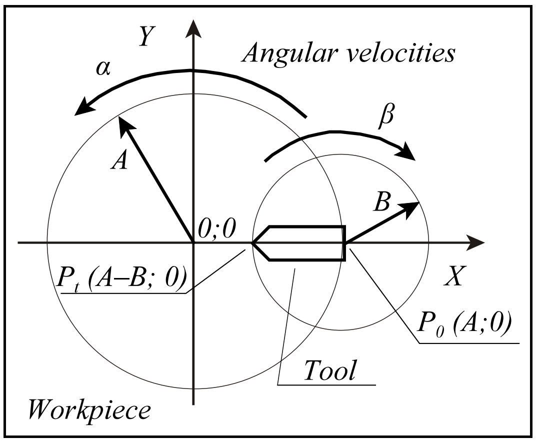


Fig. 20.1.1-1

At the moment of time t, the coordinates of the tool nose P_t (x_t ; y_t) will be the following:

$$x_t = A\cos\alpha t - B\cos(\beta-\alpha)t$$
$$y_t = A\sin\alpha t + B\sin(\beta-\alpha)t$$

Assuming that the ratio between revolutions of the tool and the workpiece is 2:1, then β=2α. Substituting this into the equations above yields:

$$x_t = A\cos\alpha t - B\cos\alpha t = (A - B)\cos\alpha t$$
$$y_t = A\sin\alpha t + B\sin\alpha t = (A + B)\sin\alpha t$$

These are the equations of an ellipse, the length of

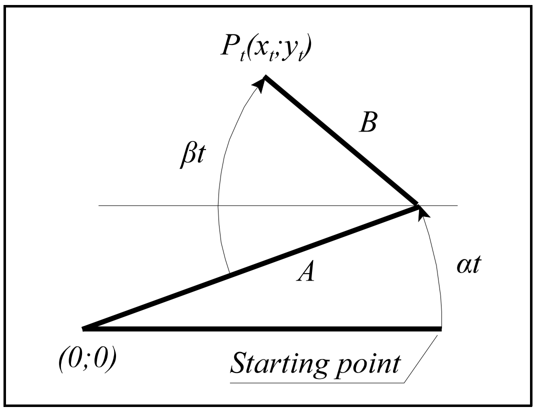


Fig. 20.1.1-2

the major axis of which is A+B and the length of the minor axis of which is A–B.

If the tools are located at 180°or 120° from each other, a square or a hexagon, respectively will be turned provided that the ratio between revolutions of the tool and the workpiece is 2:1.

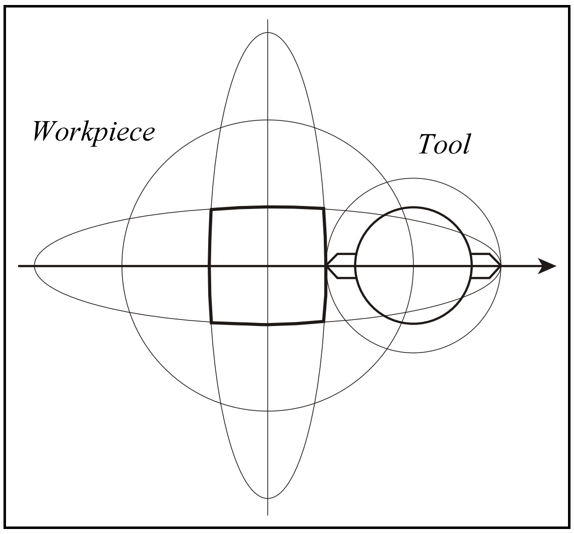


Fig. 20.1.1-3

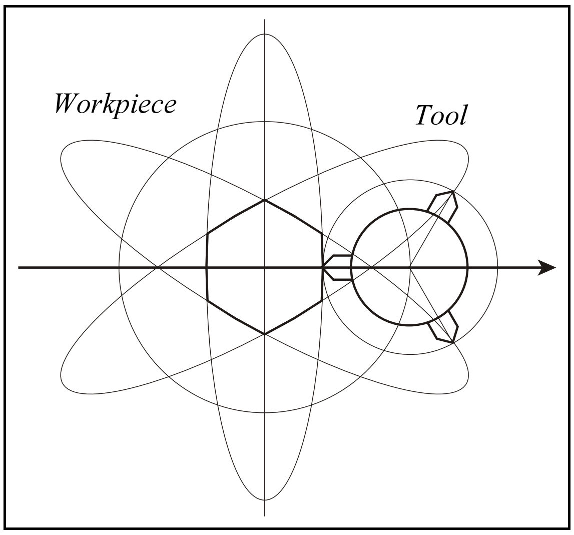


Fig. 20.1.1-4

In the cases of other revolution ratio, the curves produced will be different from ellipse, but even they will approximate the sides of the polygon well enough.

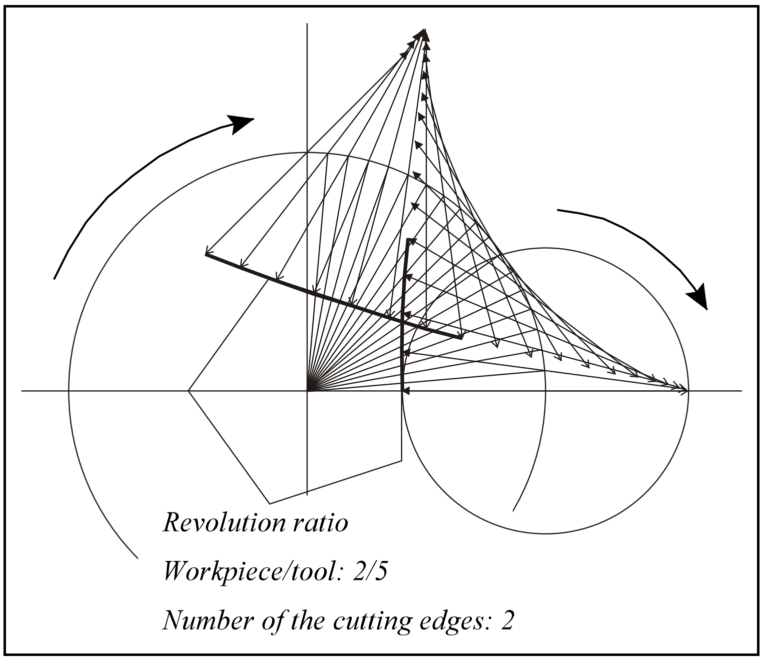


Fig. 20.1.1-5

20.1.2 Programming the Polygonal Turning (G51.2, G50.2)

The command

  G51.2 P_ Q_ R__

switches the function of the polygonal turning on.

The addresses
P and the **Q**
are those ones, at which the ratio between revolutions of the workpiece and the tool can be set:

  P/Q = revolutions of the workpiece / revolutions of the tool

If, for example, the ratio 1:2 is to be applied,

  G51.2 P1 Q2-t

will have to be programmed
At the address ***P*** always ***positive integer number*** must be given.
At the address ***Q*** either ***positive or negative integer number*** can be given. In the case of giving negative number, the tool will rotate in the direction opposite to the direction of workpiece rotation.
The addresses P and Q must always be given.

Filling the address

  R (phase shift in degree)

is ***optional***. At the address R the phase shift, i.e. the ***phase shift*** of the zero pulse of the tool spindle relative to the zero pulse of the workpiece spindle can be given ***in degree***. If the address R is not programmed, the zero pulses of the tool spindle and the workpiece spindle will coincide in each revolution Q/P of the tool spindle. The command G51.2 can be issued in the course of polygonal turning too, if the zero pulse of the tool spindle is to be shifted:

  G51.2 R_

If, for example, a square is to be turned using a single-edged tool, two opposite sides will be machined as first step using the command

  G51.2 P1 Q2.

The distance between the zero pulses of the two spindles is 0. If, after this, the command

  G51.2 R90

is programmed, machining the other two opposite sides will be possible, rotating the edge of the tool by 90° relative to the workpiece.

In the case of polygonal turning, both the workpiece spindle and the tool spindle must be equipped with an encoder, and the position control loop must be closable on the both spindles. During polygonal turning, ***the maximum speed of the spindles*** is limited by ***the rapid traverse rate*** set for the spindle axis. Generally, this value is smaller than the maximum spindle speed. In the case of polygonal turning, the ***master spindle rotating the workpiece*** is distinguished from the ***slave spindle rotating the tool***. ***Always the slave spindle has to be synchronized to the master spindle***.

The master spindle of the polygonal turning is always the spindle designated as the main spindle in the channel. The method of selection of the main spindle is determined by the builder of the machine tool, and it is contained in the manual of the given machine.

The slave spindle of the polygonal turning can be selected from the program by referencing the spindle and by function M for example; this is determined by the builder of the machine tool, and it is contained in the manual of the machine.

When polygonal turning is to be done, the process of synchronization is as follows:
– if the speed of the master spindle higher than its rapid traverse speed or if the resultant slave speed (S*Q/P) higher than the rapid traverse speed of the slave, deceleration to the appropriate speed will be executed;
– the position control loop will be closed by the control;
– the slave spindle will run up to the master speed multiplied by the ratio Q/P (S*Q/P) in the direction of rotation similar or opposite to that of the master spindle;
– the position control loop will be closed by the slave spindle;
– then, the control moves the zero pulse of the slave spindle to the zero pulse of the master spindle, or to a distance from the zero pulse of the master spindle equal to the phase shift given at the address R in degree.

The command

G50.2

Switches the polygonal turning off.
The command G51.2 and the command G50.2 have to always be given in separate blocks.

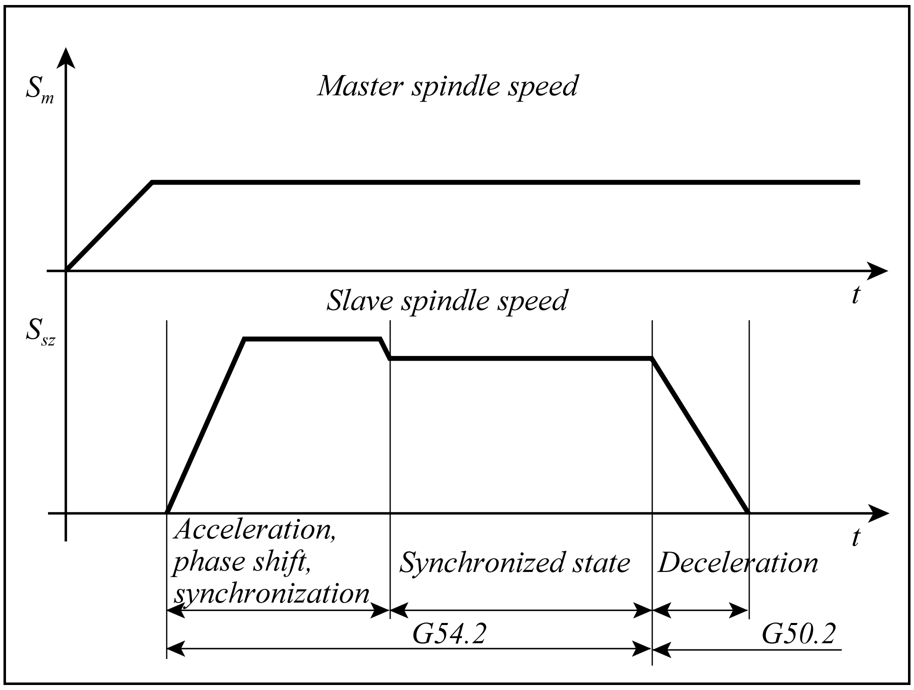


Fig. 20.1.2-1

The effect of synchronization is similar to that of the spindle zero pulse in the case of threading. So long as the workpiece is not removed from the chuck and the angle position of the cutting tool in the tool spindle is not changed, the motion of the workpiece and the motion of the tool remains synchronous and the same surface can be passed many times for the purpose of roughing and then finishing, for example. Likewise, synchronization can be switched off by the command G50.2 and then the polygon surface turned previously can be passed in the same way, provided that the programmed ratio between revolutions is the same as it was before.

☞ *Attention!*
When programming polygonal turning, take care that the speed resulting for the tool spindle

$n_{tool\ spindle}$ = S·(Q/P) never exceeds the maximum value permitted for it.

Synchronous run can be switched off
– by emergency stop;
– by servo loop errors.

Example
A hexagon is to be turned. The workpiece is clamped in the spindle S1, the tool is rotated by the spindle S3. The functions M given in the example below are occasional, they can be different from machine to machine.

Machining using three-edged tool:
```
...
M31 (DESIGNATING THE SPINDLE S1 AS MAIN SPINDLE)
S1=1000 M3
S3=0 M23 (STARTING THE SPINDLE S3 FOR POLYGONAL TURNING)
G0 X20 Z1
G51.2 P1 Q-2 (SWITCHING THE POLYGONAL TURNING ON)
X15
G1 Z-10 F0.5
G0 X30
Z1
G50.2
S1=0 M5 (STOPPING THE SPINDLE S1)
S3=0 M5 (STOPPING THE SPINDLE S3)
...
```

Machining using single-edged tool:
```
...
G0 X20 Z1
G51.2 P1 Q-2 (SWITCHING THE POLYGONAL TURNING ON)
X15
G1 Z-10 F0.5
G0 X30
Z1
G51.2 R120 (PHASE SHIFT BY 120 DEGREES)
X15
G1 Z-10 F0.5
G0 X30
Z1
G51.2 R240 (PHASE SHIFT BY 240 DEGREES)
X15
G1 Z-10 F0.5
G0 X30
Z1
G50.2
```

```
...
```

20.2 Gear Hobbing (G81.8)

Gear hobbing is described in the book NCT2xxM Control for Milling Machines and Machining Centers Programmer's Manual.

20.3 Synchronous Control of Axes

In the course of machining a workpiece, it could be necessary to move two axes synchronously. The axes can be either in the same channel or in different channels.
The axis for which the data are given in the part program is called ***master*** axis.
The axis which moves in synchronism with the master axis is called ***slave*** axis**.** So long as an axis is synchronous slave axis, this axis cannot be moved either from program or manually.
Synchronous moving is initiated by the PLC program by execution of a code M, for example.

Parameters and PLC flags used in the case of synchronous moving

If two axes are to be linked together for synchronous moving, it will ***have to be specified in the parameter*** N2101 Synchronous Master ***belonging to the slave axis which axis is its master axis***.
The master axis can be either in the same channel or in a different channel. The master axis can have several slave axes. A slave axis can also be the master axis of another axis.
In the case of synchronous control of axes, it is only the master axis that can be programmed or for which manual motion command (jog, handwheel) can be issued.

Synchronous control is requested by the PLC setting the flag AP_SYNCR belonging to the slave axis to1. The PLC waits until the control acknowledges the request through the flag AN_SYNCA. From that moment, displacements of the master axis will be received by the slave axis too.
This flag can ***usually be switched on or off by functions M.***
For example:

```
...
M41 (synchronous moving on)
...
M40 (synchronous moving off)
...
```

Hereafter, the code pair M40, M41 is used in this manual for switching synchronization on and off. In the case of a given machine, the code of ***the function and operation description should be asked the builder of the machine tool for.***

Moving in synchronism can only start strictly ***by the function*** for waiting and ***of buffer emptying***.
If moving in synchronism occurs ***between two channels***, in the case of the other channel a code for waiting will have to be programmed:

Program of the channel 1

```
...
M502 P12
M41 (synchrony on)
M503 P12
...
M504 P12
M40 (synchrony off)
M505 P12
...
```

Program of the channel 2

```
...
M502 P12
(waiting for synchrony
on)
M503 P12
...
M504 P12
(waiting for synchrony
off)
M505 P12
...
```

Another condition of requesting for synchronous moving is the existence of valid reference point both on the master axis and on the slave axis.

In the case of moving in synchronism, ***the motion direction of the slave axis*** can be the same as that of the master axis, but it can be different too. The motion direction of the slave axis can be set at the bit #0 MSY of the parameter N2102 Synchron Config which bit belongs to the slave axis. If the value of the bit of the parameter

=0: both the master axis and the slave axis move in the same direction;

=1: both the master axis and the slave axis move in opposite direction.

Example of moving the axes in synchronism

To illustrate moving axes in synchronism, let us have a two-channel engine lathe.

A bar feeder places the workpiece in the spindle S1 being in the channel 1 where the front of the workpiece will be machined. After receiving the workpiece, its end will be machined in the spindle S2 being in the channel 2.

In the channel 1, the workpiece together with the spindle S1 is moved by the axis Z_1. The turning tools T01, ..., T06 being in vertical position are moved by the axes X_1, Y_1.

In the channel 2, the spindle S2 together with the workpiece moves in the directions X_2, Z_2.

While in the spindle S1 the front of the succeeding workpiece is machined, in the spindle S2 the end of the previous workpiece will be processed.

Let us assume that both ends of the workpiece are to be machined in the same way.

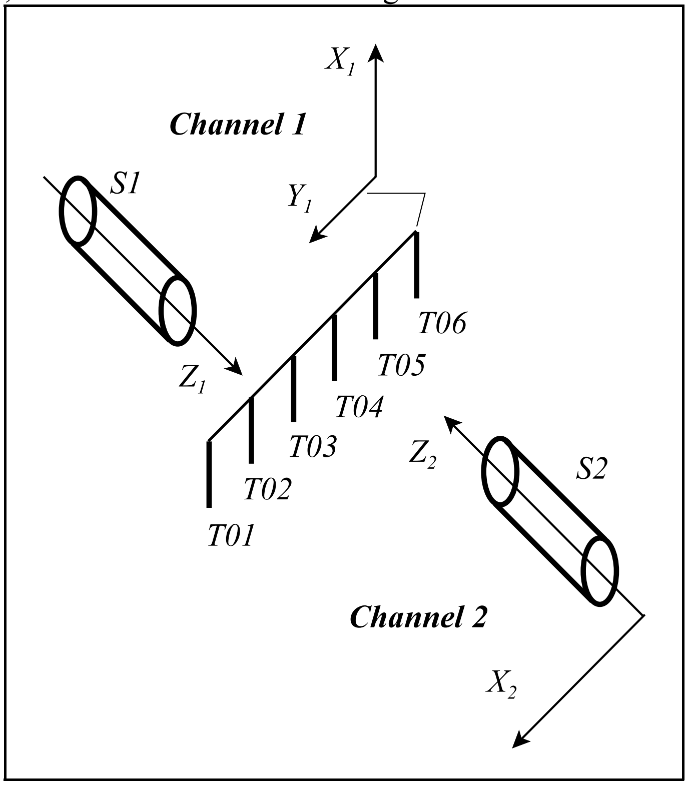


Fig. 20.3-1

Let there be tools of the same length and of the same rounding radius in the tool holders T01 and T02. If, in the channel 1, the axes X_1, Y_1 are positioned in front of the tool

T01 and , in the channel 2, the axis X_2 is positioned in front of the tool T02, ***both ends of the workpiece can be machined simultaneously too, by moving the axes Z_1 and Z_2 in synchronism***.
Let the axis Z_2 of the channel 2 be the synchronous slave of the axis Z_1 of the channel 1, what is designated in parameter.
Let us assume that the code

M41

links the axes Z_1 and Z_2 together for moving in synchronism and the spindles S1 and S2 together for rotating in synchronism,
and that the code

M40

stops the synchronous operation.
The codes M41 and M40 has to always be given in the channel 1.
Prior to machining, the workpiece being in the spindle S1has to be positioned in front of the tool ***T01*** along the axes X_1, Y_1, Z_1 in the channel 1, while the axes X_2, Z_2 has to be positioned in front of the tool ***T02*** in the channel 2. Then the synchronous cutting follows. The axes Z_1 and Z_2 moves in synchronism. The axis X_2 stands, and the end of the workpiece being in the spindle S2 is machined by the motion of the axes X_1 and Z_2. The overhang and the radius of rounding of the tool T01 and T02 have to be the same.

In the two channels, the part program is the following:

```
 The program running in the channel 1       The program running in the channel 2
...                                        ...
T101                                       T202
G0 X100 Y0 Z10                             G0   X0   Z10   (synchronous
                                           position)
M501 P12 (waiting)                         M501 P12 (waiting)
M41 (Z1, Z2 synchrony on)
M502 P12 (waiting)                         M502 P12 (waiting)
G42 G0 X0 Z2                               (the channel 2 does not work)
G1 Z0 F2
X10 ,C1
Z-5
X15
G40 G0 X100 Z10
M40 (synchrony off)
M503 P12 (waiting)                         M503 P12 (waiting)
... (normal machining)                     ... (normal machining)
```

<u>Parking the axes in the case of moving the axes in synchronism</u>

It could be necessary that a program written with the use of moving in synchronism runs only on one side, either on the master side or on the slave side; and not to write a new program.
In this case, the side not intended to be moved has to be ‘parked’. Either the slave side or the master side can be parked. In such a case too, the control calculates appropriate displacements both for the master axis and for the slave axis, but it does not issue motion command for the parking side.

Parking an axis participating in synchronous moving is requested by the PLC setting the flag of request for parking AP_PARKR to1. The control acknowledges the acknowledgment of the request for parking by setting the acknowledging flag AN_PARKA to1.
Parking can be initiated ***from push-button or from function M***. In the case of a given machine, ***description of parking should be asked the builder of the machine tool for.***

20.4 Interchanging of Axes

In the course of machining a workpiece, it could be necessary to interchange two axes. The axes can be either in the same channel or in different channels.
The axis with which the slave axis is interchanged is called ***master*** axis.
The axis which initiates the interchanged is called ***slave*** axis**.**
Interchanging the axes is initiated by the PLC program by execution of a code M, for example.

Parameters and PLC flags used in the case of interchanging the axes

If two axes are to be interchanged, it will ***have to be specified in the parameter*** N2104 Composit Axis ***belonging to the slave axis which axis is its master axis***.
The master axis can be either in the same channel or in a different channel.
In the case of interchanging the axes, both the master axis and the slave axis can be programmed, or for them manual motion command (jog, handwheel) can be issued.

Interchanging the axes is requested by the PLC setting the flag AP_MIXR belonging to the slave axis to1. The PLC waits until the control acknowledges the request through the flag AN_MIXA. From that moment, the two axis are interchanged.
This flag can ***usually be switched on or off by functions M.***

For example:

```
...
M42 (axis interchange on)
...
M40 (axis interchange off)
...
```

Hereafter, the code pair M40, M42 is used in this manual for switching the axis interchange on and off. In the case of a given machine, ***the code of the function and operation description should be asked the builder of the machine tool for.***

Interchanging the axes can only start strictly ***by the function*** for waiting and ***of buffer emptying***.
If interchanging the axes occurs ***between two channels***, in the case of the other channel a code for waiting will have to be programmed:

Program of the channel 1

```
...
M502 P12
M42 (axis interchange on)
M503 P12
...
M504 P12
M40   (axis   interchange
off)
M505 P12
...
```

Program of the channel 2

```
...
M502 P12
(waiting for interchange)
M503 P12
...
M504 P12
(waiting for deleting the
interchange)
M505 P12
...
```

After interchanging the axes, ***the motion direction of the slave axis*** can be its original one, but it can be opposite, too. The motion direction of the slave axis can be set at the bit #0 MMI of the parameter N2105 Composit Config which bit belongs to the slave axis. If the value of the bit of the parameter

=0: the slave axis moves in the original direction after interchanging the axes;

=1: the slave axis moves in the direction opposite to the original one after interchanging the axes.

After interchanging the axes, ***the motion direction of the master axis*** can be its original one, but it can be opposite, too. The motion direction of the master axis can be set at the bit #0 MMI of the parameter N2105 Composit Config which bit belongs to the master axis. If the value of the bit of the parameter

=0: the master axis moves in the original direction after interchanging the axes;

=1: the master axis moves in the direction opposite to the original one after interchanging the axes.

After interchanging the axes, ***the program can be written both on the real and the virtual*** (mirrored) ***sides***.

The programming on the virtual side can be set at the bit #2 MCO of the parameter N2105 Composit Config which bit belongs to the slave axis. If the value of the bit of the parameter

=0: the slave axis has to be programmed on the real side after interchanging the axes;

=1: the slave axis has to be programmed on the virtual side after interchanging the axes.

The programming on the virtual side can be set at the bit #2 MCO of the parameter N2105 Composit Config which bit belongs to the master axis. If the value of the bit of the parameter

=0: the slave axis has to be programmed on the real side after interchanging the axes;

=1: the slave axis has to be programmed on the virtual side after interchanging the axes.

After interchanging ***the axes, the axes will have their original zero-point offsets and length compensation***. For example, if the axes X_1-X_2 and the axes Z_1-Z_2 are changed between the two channel, the length compensation set for the axis X_1 will go to the channel 2 and vice versa. Therefore, it is advisable to call a new zero-point offset measured after the axis change and a new length compensation.

Example of interchanging the axes

Let there be ***two turret on a two-channel machine.***

The ***first turret is managed by the channel*** 1and it can be moved along the axes X_1, Z_1. The ***second turret is managed by the channel 2*** and it can be moved along the axes X_2, Z_2.

Because of receiving the workpiece, the subspindel S2 in the channel 2 can be moved by the axis W_2.

Being in the spindle S1 the one

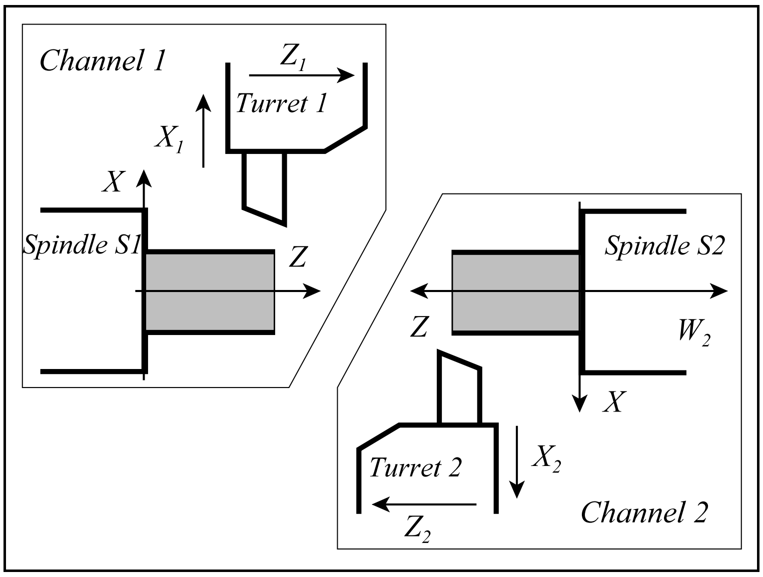


Fig. 20.4-1

side of the workpiece, while being in the spindle S2 the other side of the workpiece is machined. In both channels, identical tools could be necessary for machining.

It could come up in the case of machining more complex workpiece using tools of many kinds, that there are no so many tool stations in the one or in the other turret required for mounting identical tools in both turrets. In this case, if in one of the channels the tool stored in the other turret is required, ***it will be necessary to interchange the turrets***, i.e. the axes X_1, Z_1 and X_2, Z_2 ***between the channels***.
Then, machining can be executed in both channels.

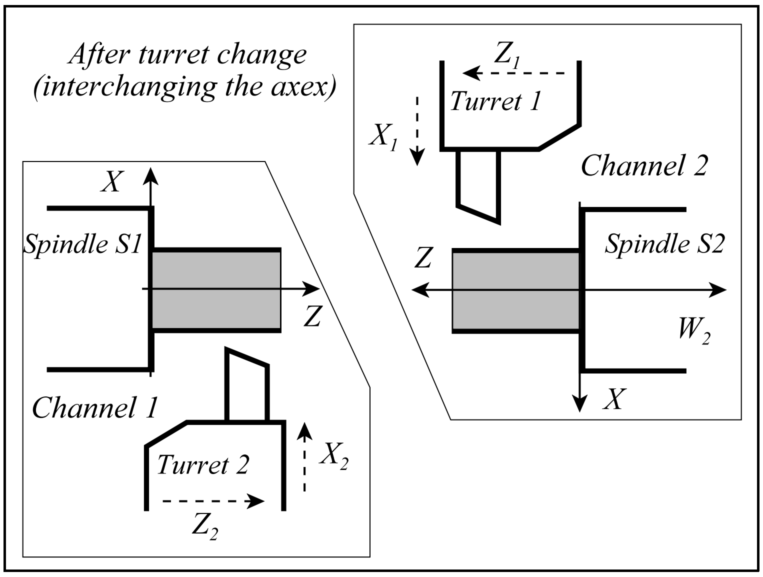


Fig. 20.4-2

As it can be seen in the figure, ***the direction of both the axes X_1, Z_1 and X_2, Z_2 has to be reversed*** writing 1 at the bit MMI in order that the directions will adapt to coordinate directions given in the channel.

Then, it has to be decided whether the program will be written on the real or virtual side. If, after interchanging the axes, the program is written ***on the real side*** either in the channel 1 or in the channel 2, ***the tool will approach the workpiece during motion of the axis X in the positive direction***, and the tool will move away from the workpiece during motion of the axis X in the negative direction. Programming has to be done contrary to the programming done before interchanging the axes!

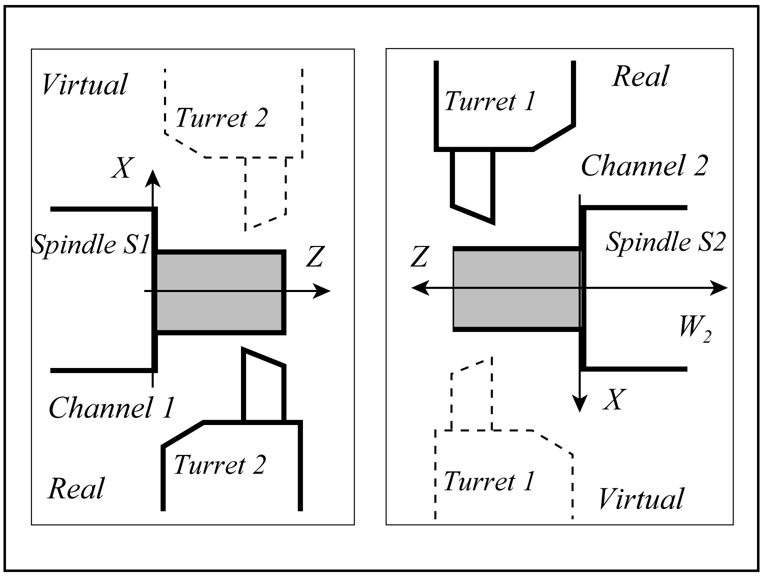


Fig. 20.4-3

If the program is to be written ***on the virtual side*** as it was done before interchanging the axes, 1 will have to be written at the bit MMO belonging to the axes X_1 és az X_2.
In this case, ***the program*** written in such a way ***will be automatically mirrored through the axis Z by the control***.

This figure shows a workpiece being in the channel2and to be machined after interchanging the axes by the tool in the turret being moved by the axes X_1, Z_1.
The real absolute position of the tool in the coordinate system fixed to the workpiece is

$x_2 = -120$, $z_2 = 80$.

The following two examples illustrate program writing on the ***real side*** at the parameter position
$MCO_{X2}=0$,
and program writing on the ***virtual side*** at the parameter position
$MCO_{X2}=1$.
Writing the program on the virtual side looks like as if interchanging the axes had been passed unmarked and the machining would be executed using the original turret.

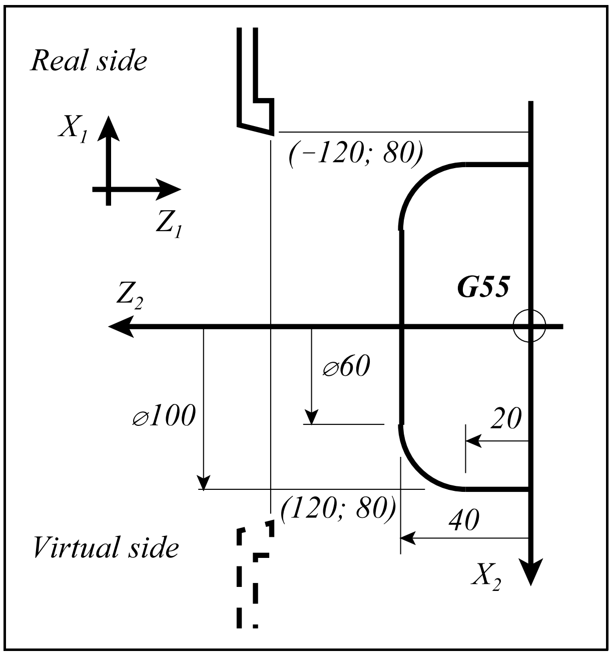


Fig. 20.4-4

At the tool nose radius compensation, ***the tool position code Q has to be given according to the side the program was written on***. In the following example, there has to be given Q2 on the real side, and Q3 on the virtual side.

Real side, MCO_{X2}=0:
```
... (machining by the axes X2, Z2)
(interchanging the axes, machining by the axes X1, Z1)
M502 P12
G55 T101 (R0.8, Q2)
G0 X-120 Z80
G41 G0 X0 Z40
G1 X-60 F0.5
G2 X-100 Z20 R20
G1 Z0
G0 X-120
G40 Z80
M503 P12
...
```

Virtual side, MCO_{X2}=1:
```
... (machining by the axes X2, Z2)
(interchanging the axes, machining by the axes X1, Z1)
M502 P12
G55 T101 (R0.8, Q3)
G0 X120 Z80
G42 G0 X0 Z40
G1 X60 F0.5
G3 X100 Z20 R20
G1 Z0
G0 X120
G40 Z80
M503 P12
...
```

Certainly, the tool will always move on the real side.

Interchanging real and hypothetical axes

There is a case, when ***a real axis and an unexisting axis are to be interchanged***. Such a case is illustrated in the figure showing kinematic scheme of an engine lathe.

A bar feeder places the workpiece in the spindle S1 being in the channel 1 where the front of the workpiece will be machined. After receiving the workpiece, its end will be machined in the spindle S2 being in the channel 2.

In the channel 1, the workpiece together with the spindle S1 is moved by the axis Z_1. The turning tools T01, ..., T06 being in vertical position are moved by the axes X_1, Y_1.

In the channel 2, the spindle S2 together with the workpiece moves in the directions X_2, Z_2.

While in the spindle S1 the front of the succeeding workpiece is machined, in the spindle S2 the end of the previous workpiece is processed.

There is no axis Y_2 in the channel 2.

Let us assume that the end of the workpiece has to be machined using one of the tools of the tool group T01, ..., T06, in the channel 2.

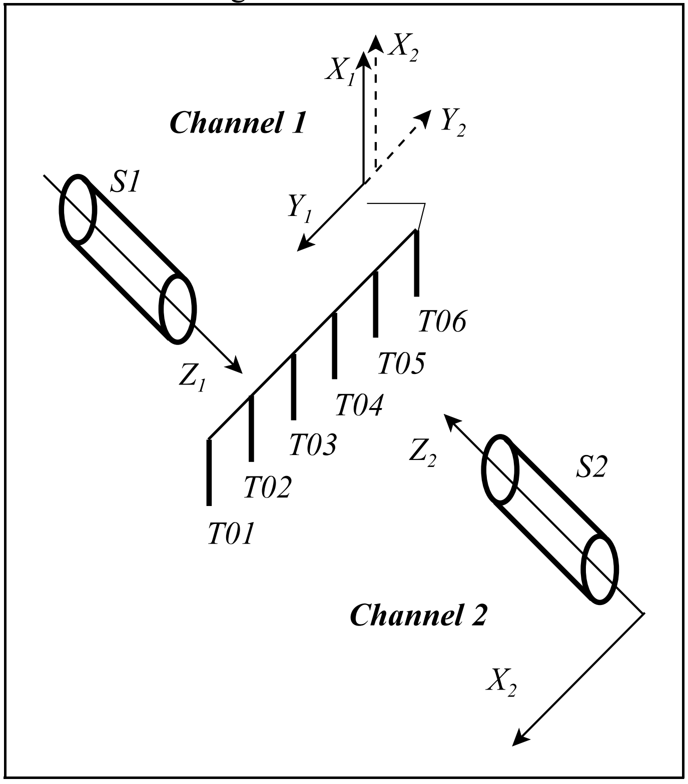


Fig. 20.4-5

In this case, ***the axes X_1, Y_1 and X_2, Y_2 have to be interchanged between the channels 1 and 2***. In order that the mechanism of interchange can pass off, ***a hypothetical axis Y_2 has to be designated in the channel 2.***

Hypothetical axis can be designated by the bit position #7 HYP=1of the parameter N0106 Axis Properties. ***A hypothetical axis has a name*** (in this case it is Y) and ***a number***, but ***physical axis output and input*** from the servo parameters ***are not assigned to it***, i.e. encoder input and drive output do not belong to it.

After interchanging the axes, in this case in the channel 2, the axis Y can be referred both from program and by manual moving.

It can be seen in the figure that the positive direction of the axis Y_2 is opposite to the positive direction of the axis Y_1 in order that the coordinate system X_2, Y_2, Z_2 will be right-handed. For this purpose, 1 has to be set at the bit #0 MMI of the parameter N2105 Composit Config which bit belongs to the axis Y_2.

Since, after interchanging the axes, the hypothetical axis got to the channel1, the axis Y cannot be referred either from program or by manual moving. Likewise, the axis X also cannot be referred in the channel 1after changing the axis X would be moved by the spindle S2.

20.5 Superimposed Control of Axes

In the course of machining a workpiece, it could be necessary to add motion of an axis to motion of another axis. It is called superimposed moving the axes. The axes can be either in the same channel or in different channels.
The axis, motion of which is added to motion of the slave axis, is called ***master*** axis.
The axis, to motion of which the motion of master is added, is called ***slave*** axis**.** Both the master axis and the slave axis can be moved from program.
Superimposed moving is initiated by the PLC program by execution of a code M, for example.

Parameters and PLC flags used in the case of superimposed moving the axes

If two axes are to be linked together for superimposed moving, it will ***have to be specified in the parameter*** N2107 Superimposed Master ***belonging to the slave axis which axis is its master axis***.
The master axis can be either in the same channel or in a different channel. The master axis can have several slave axes. A slave axis can also be the master axis of another axis.
In the case of superimposed control of axes, both the master axis and the slave axis can be programmed or manual motion command (jog, handwheel) can be issued.

Superimposed control is requested by the PLC setting the flag AP_SPRPNR belonging to the slave axis to1. The PLC waits until the control acknowledges the request through the flag AN_SPRPNA. From that moment, displacements of the master axis will be received by the slave axis too, and displacements of the master axis will be added to the displacements of the slave axis.
This flag can ***usually be switched on or off by functions M.***
For example:

```
...
M43 (superimposed moving on)
...
M40 (superimposed moving off)
...
```

Hereafter, the code pair M40, M43 is used in this manual for switching superimposing on and off. In the case of a given machine, the code of ***the function and operation description should be asked the builder of the machine tool for.***

Superimposed moving can only start strictly ***by the function*** for waiting and ***of buffer emptying***.
If superimposed moving occurs ***between two channels***, in the case of the other channel a code for waiting will have to be programmed:

Program of the channel 1

```
...
M502 P12
M43 (superimposing on)

M503 P12
...
M504 P12
M40 (superimposing off)

M505 P12
...
```

Program of the channel 2

```
...
M502 P12
(Waiting for
superimposing)
M503 P12
...
M504 P12
(Waiting for
superimposing)
M505 P12
...
```

Another condition of requesting for superimposed moving is the existence of valid reference point both on the master axis and on the slave axis.

In the case of superimposed moving, the motion of the master axis can be added or subtracted ***to or from the motion of the slave axis***. It can be set at the bit #0 MSU of the parameter N2108 Superimposed Config which bit belongs to the slave axis, whether the master axis will be added or subtracted to or from the motion of the slave axis. If the value of the bit of the parameter

=0: the motion of the master axis will be added to the motion of the slave axis;

=1: the motion of the master axis will be subtracted from the motion of the slave axis.

Example of superimposed moving the axes

Let there be two turret on a two-channel machine.

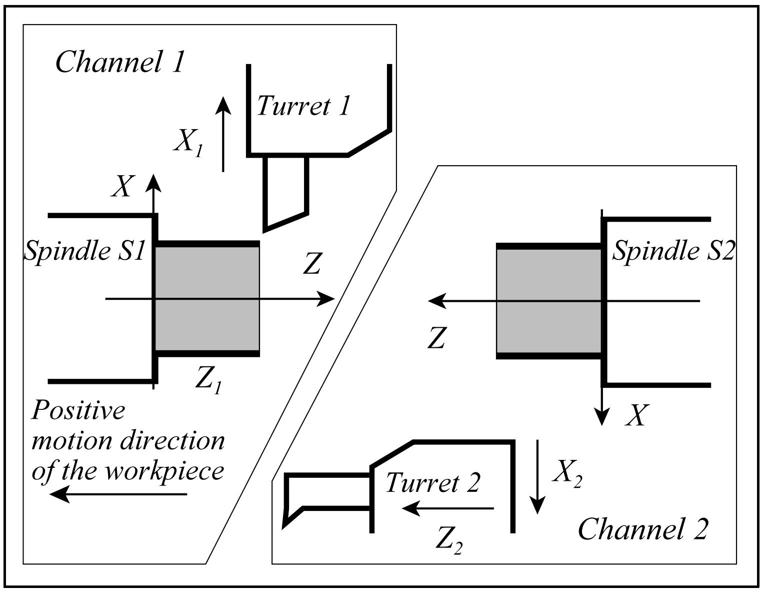


Fig. 20.5-1

The first turret is managed by the channel 1and it can be moved along the axis X_1, ***while the spindle S1 together with the workpiece is moved by the axis Z_1***.

The second turret is managed by the channel 2 and it can be moved along the axes X_2, Z_2. The subspindle in the channel 2 stands. Being in the spindle S1 the one side of the workpiece, while being in the spindle S2 the other side of the workpiece is machined.

Let us assume that ***the outside surface of the workpiece being in the spindle S1 will be machined by the tool being in the turret 1, while the inside surface of this workpiece will be machined by the tool being in the turret 2***.

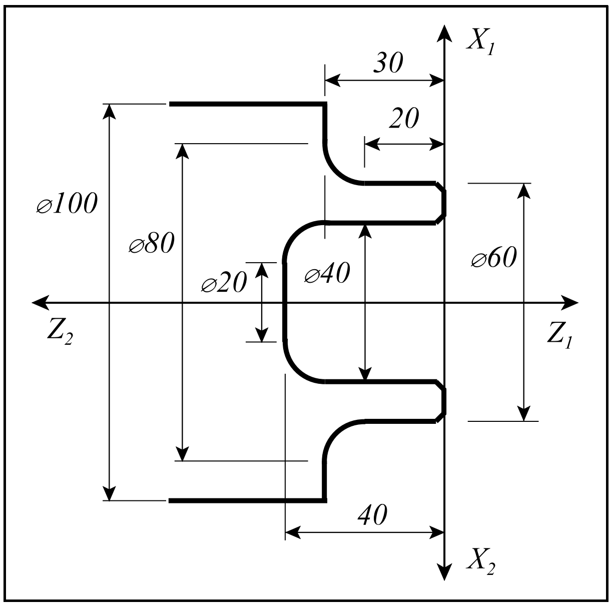


Fig. 20.5-2

The program running in the channel 1 moves the axes X_1, Z_1. At the same time, the program running in the channel 2 moves the axes X_2, Z_2. Since ***the axis Z_1 moves the workpiece being in the spindle S1*** right and left, ***the displacement of the axis Z_1 has to be added to the displacement of the axis Z_2*** in order that the position of the tool being in the turret 2 will not change relative to the workpiece because of motion of the axes X_2, Z_2.

In this arrangement ***Z_1 is the master axis and Z_2 is the slave axis***. ***If the master axis Z_1 moves in positive direction, the slave axis Z_2 will have to be moved in the positive***

direction too in order that the position of the tool being in the turret 2 will not change relative to the workpiece being in the spindle S1. For this, the bit position #0 MSU=0 has to be set in the parameter N2108 Superimposed Config for the slave axis.
The programs for the workpiece illustrated in the drawing are the following:

The program running in the channel 1 — **The program running in the channel 2**

```
...                               ...
T101 (tool position Q3)           T2121 (tool position Q1)
G55 G90 G40 G0 X120 Z10           G55 G90 G40 G0 X120 Z-10
M3 S1=1000
M501 P12                          M501 P12
M43 (superimposed moving)
M502 P12                          M502 P12
G42 G0 X50 Z2                     G42 G0 X60 Z-5
G1 X60 ,C3 F0.6                   G1 Z0 F0.8
Z-20                              X40 ,C1
G2 X80 Z-30 R10                   Z30
G1 X100                           G2 X20 Z40 R10
G40 G0 Z10                        G1 X0
                                  G40 G0 Z-10
                                  X120
M503 P12                          M503 P12
M40 (normal machining)
M504 P12                          M504 P12
G54 G0 X200 Z20                   G54 G0 X150 Z10
...                               ...
```

20.6 Changing the Axis Direction

In the control, it is possible to change the movement direction of an axis, from the PLC program: the positive direction command will cause a negative movement and vice versa.

It is the PLC that requests changing the axis direction by setting the flag AP_MIRR belonging to the given axis to 1. The PLC waits until the NC acknowledges the request through the flag AN_MIRA. From then on, the movement direction of the axis will be opposite to the direction set in parameter.

In general, this flag ***can be switched*** on or off ***by the functions M***.
In the case of a given machine, please ***ask the builder of the machine for the code and operation of the function***.
Changing the axis direction can only start strictly by the ***function of buffer emptying***.

Changing the axis direction does not affect the direction of movement and the positions of the positionings (G53) programmed in the machine coordinate system and of the moving to the reference point (G28, G30 P).

Example of changing the axis direction

Now the turret T1 with its tools machines the front side of the workpiece being in the spindle S1. The tool moves relative to the workpiece, in the directions X-Z. Then, the NC puts the workpiece over to the spindle S2 being in the turret T1 (see the Figure). After revolving the turret, the spindle S2 locates against the gang-type tool group T2 used for machining the rear part of the workpiece. In this case, the workpiece being in the spindle S2 moves relative to the tool group T2.

In order that the programs can be written in conventional way (the tool moves away from the workpiece in the positive direction of motion) and that the coordinate system remains right-handed, namely that the circle directions, direction of the radius compensation etc. do not change, the motion direction of the X and Z axes has to also be reversed.

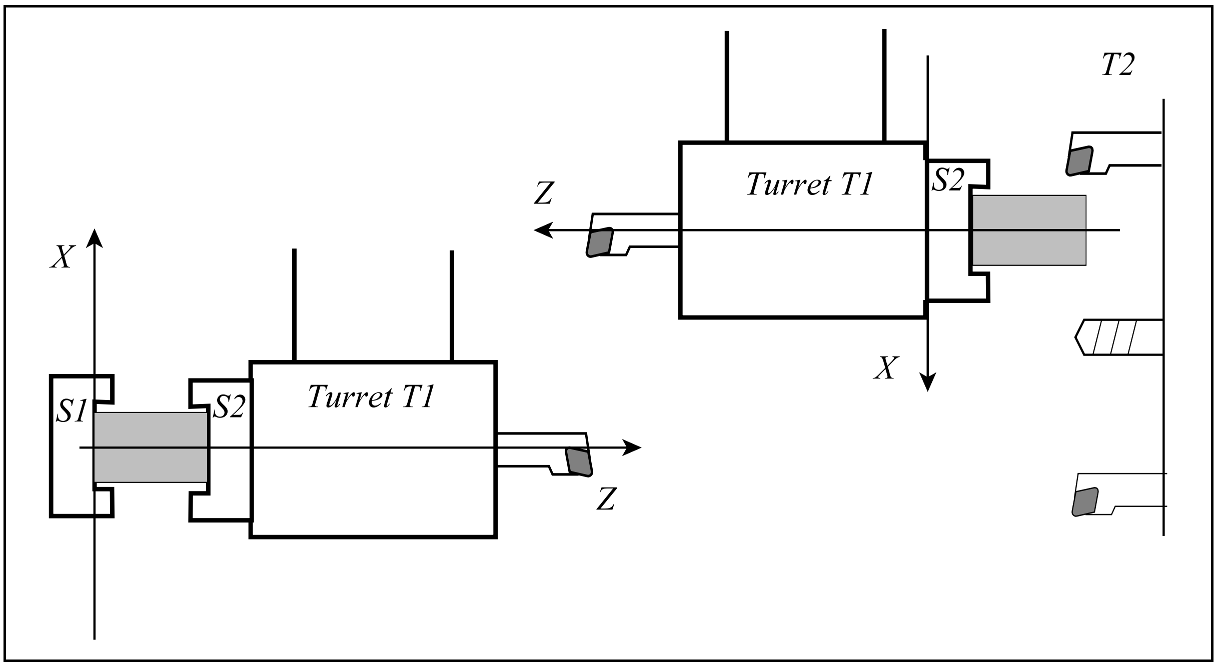


Fig. 20.6-1

Let, for example, M44 be the function M that executes changing the axis direction, and the code M40 cancels the change of direction.

```
...
G54 T101
...
M44 (change of direction on the axes X-Z)
G55 T2020 (new workpiece zero point and length compensation)
...
M40 (cancelling the change of direction on the axes X-Z)
G54 T101 (new workpiece zero point and length compensation)
...
```

After changing the axis direction, change-over to the coordinate system fixed to the workpiece that was measured after the change of direction, and to the length compensation belonging to the group T2. The situation is the same after canceling the change of direction, too.

20.7 Managing Non-perpendicular Axes

If the movement of one axis on a machine is not perpendicular to the other, but ***makes an angle with respect to the right angle***, managing angular or slant axis is to be applied.

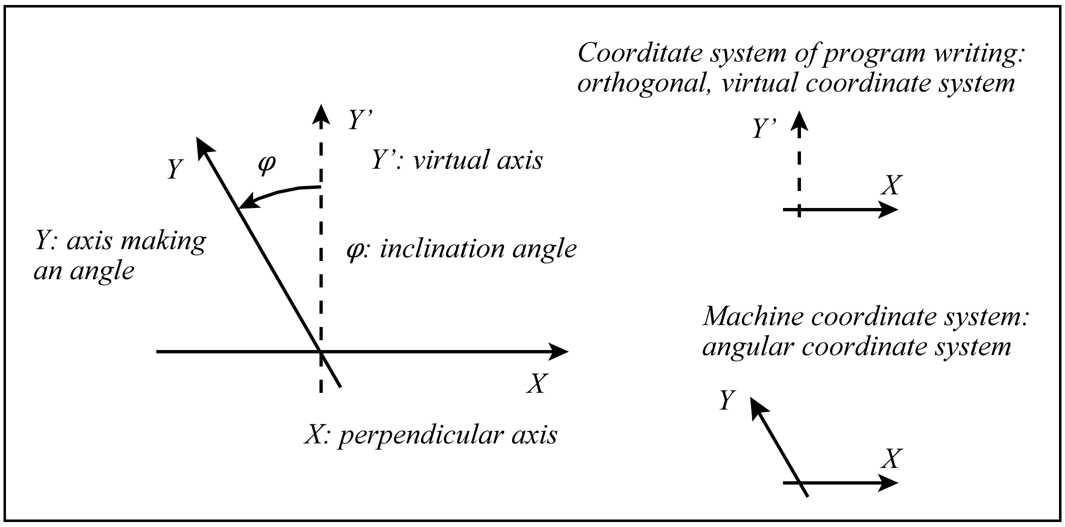

Coorditate system of program writing: orthogonal, virtual coordinate system
Y'
Y
φ
Y': virtual axis
X
Y: axis making an angle
φ: inclination angle
Machine coordinate system: angular coordinate system
X
X: perpendicular axis
Y
X

Fig. 20.7

In each channel, a pair of axes can be designate in parameter to manage angular axes.
An axis existing in the orthogonal coordinate system is defined as ***perpendicular axis***, and an axis not perpendicular to it is defined as ***angular or slant axis.***
The part program is written in an ***orthogonal virtual coordinate system***, in the Figure it is the X-Y' coordinate system, where ***Y' is the virtual axis.***
The function that manages the slant axis automatically converts the program written in this way into the coordinate system that makes angle.
The position is displayed in the orthogonal virtual coordinate system.
Using a PLC display, managing the slant axis can be switched on and off from the program by means of M functions, for example.

Displacement

Let

$\mathbf{X_a}$ mark the current displacement along the perpendicular axis,
$\mathbf{Y_a}$ mark the current displacement along the slant axis,
$\mathbf{X_p}$ mark the programmed displacement along the perpendicular axis,
$\mathbf{Y'_p}$ mark the programmed displacement along the virtual axis,
φ mark the angle included between the virtual axis and the slant axis.

The relationship between the programmed and the current displacements is as follows:

$$Y_a = \frac{Y'_p}{\cos\varphi}$$

$$X_a = X_p + cY'_p \, tg\varphi$$

where

c=1, if the perpendicular axis (X_p-t) is given in radius,
c=2, if the perpendicular axis (X_p-t) is given in diameter.

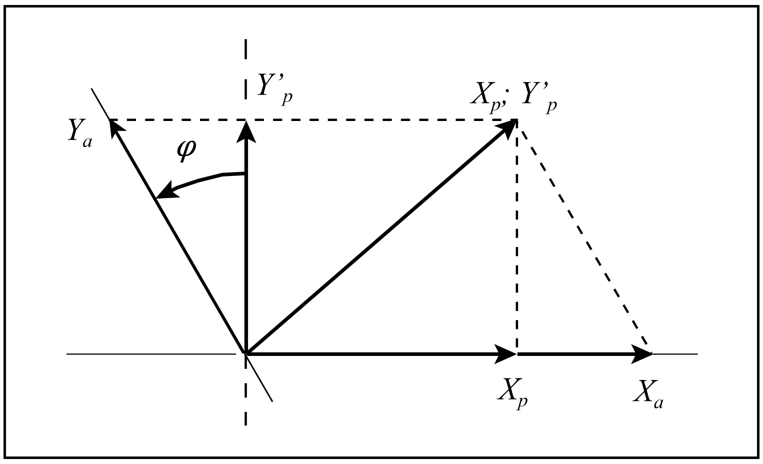


Fig. 20.7

Feed

The programmed feed must always be given in the orthogonal coordinate system.
Let

F_p mark the programmed feed,
F_{ax} mark the feed component along the perpendicular axis,
F_{ay} mark the feed component along the slant axis,
φ mark the angle included between the virtual axis and the slant axis.

The programmed feed will be decomposed into components valid along the real axes in accordance with the following relationship:

$$F_{ya} = \frac{F_p}{\cos\varphi}$$

$$F_{xa} = F_p + F_p tg\varphi$$

Switching off managing the slant axis

On the control, managing the slant axis can be switched on and off, using push button or M functions.
Location of the push button, and the M function is determined by the machine builder. The M function can also be used during program execution, provided the M function is designated to buffer flush.
With setting the PLC flag **CP_NOANGCR** to 1, managing the inclined axis will be switched off when the acknowledge signal **CN_NOANGCA** is switched on by the control.

Parameters for setting the managing of the inclined axis

Managing the inclined axis can be enabled by setting the **#0 ANE** bit of the parameter **N0111 Angular Axis Control** to 1**.**
The parameter **N0112 Slant Angle** can be used to specify the degree of deviation of the direction of the slant axis from the perpendicular direction.
The number of the slant axis can be specified in the parameter **N0113 Axis Number of Slant**.
Only axes existing in the given channel can be specified.

Finally, the parameter **N0114 Axis Number of Cartesian** allows you to specify the number of the perpendicular axis to which the motion of the slant axis is reflected. Only axes existing in the given channel can be specified.

Positioning and interpolation in the workpiece coordination system

Case 1: Managing the slant axis is switched on (the state of the PLC flag: CN_NOANGCA=0)
Data of the position and the feed are always specified in the ***virtual*** orthogonal ***coordinate system***.
The ***position*** is also ***displayed*** in the ***virtual coordinate system***.
The position of the perpendicular axis ***is corrected*** with the position of the slant axis by the interpolator.

Example:
```
N1 G0 G90 Y100      (positioning from P0 to the point P1)
N2 X200         (positioning from P1 to the point P2)
```

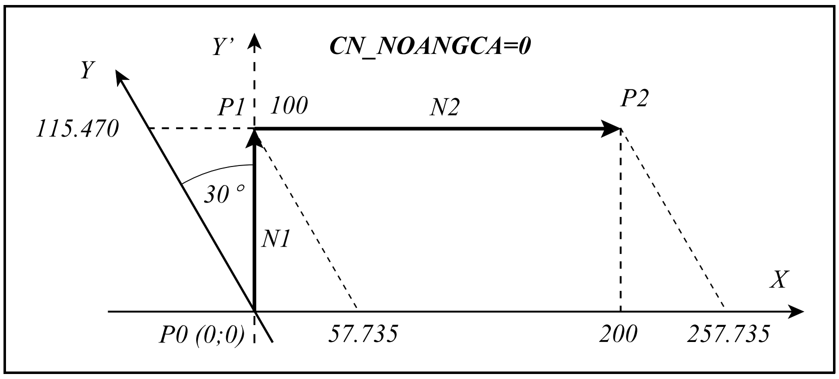


Fig. 20.7

Positions in the point P1 at the end of the block N1:

Programmed position	Position in the real coordinate system
`X0.000`	`X 57.735`
`Y100.000`	`Y115.470`

Positions in the point P2 at the end of the block N2:

Programmed position	Position in the real coordinate system
`X200.000`	`X257.735`
`Y100.000`	`Y115.470`

The above ***also*** applies to ***manual movings***.
By selecting the slant axis for jog or handwheel moving, ***the position of the perpendicular axis will be corrected with the position of the slant axis***, and the motion will be performed in the virtual coordinate system.
For example, if the incremental jog Y button is pushed and the step value is 1 mm, the axis Y and axis X will move 1.155 mm and 0.578 mm, respectively.

Case 2: Managing the slant axis is switched off (the state of the PLC flag: CN_NOANGCA=1)
Data of the position and the feed are always specified in the ***virtual coordinate system***.
The ***position*** is also ***displayed*** in the ***virtual coordinate system***.
The position of the perpendicular axis ***is not corrected*** with the position of the slant axis by the interpolator.

Example:
```
    N1 G0 G90 Y100      (positioning from P0 to the point P1)
    N2 X200        (positioning from P1 to the point P2)
```

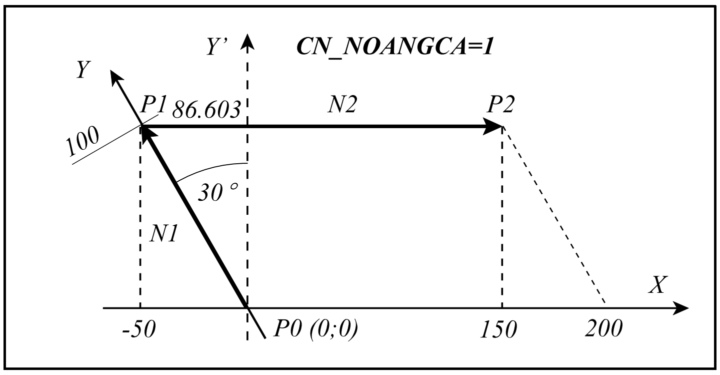


Fig. 20.7

<u>Positions in the point P1 at the end of the block N1:</u>

Programmed position	Position in the virtual coordinate system
X0.000	X-50.000
Y100.000	Y86.603

<u>Positions in the point P2 at the end of the block N2:</u>

Programmed position	Position in the virtual coordinate system
X200.000	X150.000
Y100.000	Y86.603

The above ***also*** applies to ***manual movings***.
For example, if the incremental jog Y button is pushed and the step value is 1 mm, the axis Y will move 1 mm while axis X will not move.

<u>Positioning in the machine coordinate system: G53</u>

Positioning in the machine coordinate system is not affected by the state of the flag CN_NOANGCA. Movement of the slant axis does not compensate position of the perpendicular axis.
The position data are interpreted in the real slant coordinate system.

Example:
```
     N1 G53 Y100    (positioning from P0 to the point P1)
     N2 G53 X200    (positioning from P0 to the point P1)
```

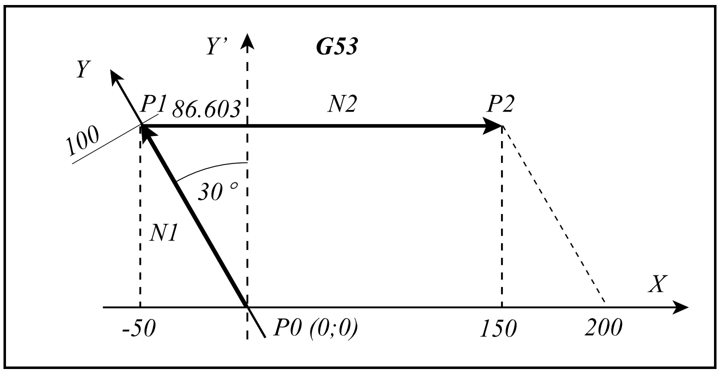


Fig. 20.7

Positions in the point P1 at the end of the block N1:

Virtual position	Machine position
X-50.000	X 0.000
Y 86.603	Y100.000

Positions in the point P2 at the end of the block N2:

Virtual position	Machine position
X150.000	X200.000
Y 86.603	Y100.000

Drilling cycles

If the slant axis is selected as the drilling axis, drilling is only possible if managing the slant axis is not switched off (CN_NOANGCA=0 state), otherwise the error message "Drilling cycle not possible with state CN_NOANGCA=1" is issued. In the case of G84.2 and G84.3, there will be interpolation between spindle and virtual axis.

Execution of the codes G28 and G30

The position of the ***intermediate points*** is always interpreted in the ***virtual coordinate system***. Depending on the status of the flag CN_NOANGCA,
the position of the perpendicular axis will be compensated by moving the slant axis (CN_NOANGCA=0) or ***there will not be compensation on the perpendicular axis*** (CN_NOANGCA=1).
The positions specified in the parameters N0200 Reference Position1, ..., N0203 Reference Position4 are always specified ***along the slant axis***.
When motion from an intermediate point to the reference point occurs, motion of the slant axis will not compensate motion of the perpendicular axis.

Manual reference point return and G28

In the case of manual reference point return, or in the case of execution of G28 when reference point return is not occurred yet, compensation of the perpendicular axis will or will not be implemented depending on the state of the flag CN_NOANGCA.

Specifying the end position data

End positions have to be specified and taken into account always along the slant axis.

Transformations

Transformations (G52, G51, G51.1, G68, G68.2) are always interpreted in the virtual coordinate system.

Zero point offset and length compensation data

Zero point offset and length compensation data for the slant axis are always stored ***according to the virtual axis***.
This is also necessary because, when measuring tool overhangs or workpiece zero point on an external device, the offsets are obtained in orthogonal coordinate system.

Measuring zero point and lenth compensation

During measurement, offsets must always be calculated from the virtual axis positions.

Macro variables

Interpretation of the macro variables is as follows:
#5001..., #5041..., #5061: poition information in the **virtual** coordinate system
#5021...: position information in the real, **slant** axis coordinate system
#5081..., #5121: length compensations in the **virtual** coordinate system
#5101...: lag in the real, **slant** axis coordinate system
#100651...: handwheel zero point offset in the **virtual** coordinate system
#5181...: distance to go in the **virtual** coordinate system
#10001...: length compensations in the **virtual** coordinate system
#5201..., 12001..., #7001..., #12061...,: zero point offsets and misalignment compensations in the **virtual** coordinate system
#5521...: dynamic zero point offsets in the **virtual** coordinate system
#5501...: the current dynamic zero point offsets in the **virtual** coordinate system

G10

When zero point offsets and compensations are entered, the data sspecified are entered in the **virtual** coordinate system.

Moving the axes in synchronism, change of axis, superimposed movings

The above moving the axes including an angle is possible if the slant axis forms a pair with another slant axis, while the perpendicular axis forms a pair with another perpendicular axis:
perpendicular axis X1- pair - perpendicular axis X2
slant axisY1 - pair - slant axis Y2.
In the case of an other setting the error message ‘Setting the slant axis synchronism, superimposed, moving, change of axis is erronous’ will be issued.

21 Measurement Functions

21.1 Skip Function (G31)

The command

G31 v (P) (F)

starts ***the motion*** to the point of coordinate ***v*** using ***linear interpolation***. The motion continues until an ***external skip signal*** (e.g. the signal of a touch probe) arrives or ***the endpoint position of the coordinates v*** is reached by the control. After arriving of the skip signal or at the programmed endpoint of the block the control will decelerate and stop.

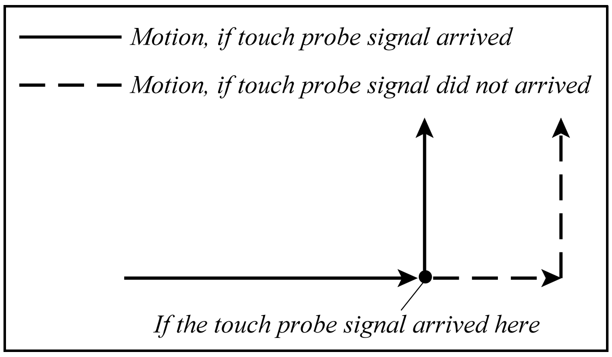

Motion, if touch probe signal arrived
Motion, if touch probe signal did not arrived
If the touch probe signal arrived here

Fig. 21.1-1

The touch probe signal among the 8 ones inputable to the control which is to be taken into account in the course of the motion, can be specified ***at the address P***:

P1: using the touch probe signal 1;
P2: using the touch probe signal 2;
...
P8: using the touch probe signal 8.

Filling the address P is not obligatory; if the address P is not filled, the touch probe signal 1 will be taken into account by the control.

The function G31 has to always be used in the state ***G94 (feed per minute)***. During the motion, ***the feed F*** will be:

– a specified on modal value, if the bit #0 SKF of the parameter N3001 G31 Config is 0;
– a value taken on from the parameter N0311 G31 Feed, if the bit #0 SKF of the parameter N3001 G31 Config is 1.

The command ***G31 is not a modal one,*** it is valid only in the block in which is programmed.

At the moment the external signal arrives, ***the positions of the axes will be stored in the following macro variables***:

#5061 or #100151 or #_ABSKP[1]: position of the axis 1;
#5062 or #100152 or #_ABSKP[2]: position of the axis 2.
...

The positions stored in the macro variable above will be the following:
– the position taken ***at the moment the signal arrives, if the external signal arrived;***
– the position of the programmed ***endpoint*** of the block G31, if the external signal ***did not arrive***.

The position data will be stored
– always ***in the coordinate system of the actual workpiece***;
– ***without taking the actual length compensation*** (G43, G44) ***into account***.

After the external signal arrives, motion will stop with deceleration. At this time, the endpoint position of the block G31deviates to a small extent from the positions stored in the variables #5061... at the moment the signal arrives, according to the feed used in the block. The endpoint positions of the block can be accessed in the variables #5001... . The next motion block will be valid from these endpoint positions.

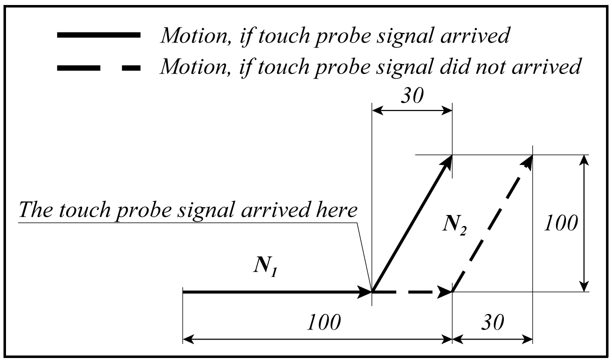
Execution of the block G31is possible only in the states ***G15, G40, G50, G50.1 G69, G94***. Otherwise, the error message '2055 Skip function in Gnn state' will be induced.

The value given at the coordinates v can be both ***incremental and absolute.*** If coordinate specification of the succeeding block is incremental, displacement will be calculated by the control from that point of the block G31where the motion stopped in the previous block.

For example:

```
N1 G31 G91 Z100
N2 X100 Z30
```

In the block N1, an incremental motion in the direction Z is started by the control. If, after arriving of the external signal, the control stops at the point with the coordinate of Z=86.7, the following incremental motions from that point will be executed in the block N2: 100 in the direction X and 30 in the direction Z.



Motion, if touch probe signal arrived
Motion, if touch probe signal did not arrived
30
The touch probe signal arrived here
N_2
100
N_1
100
30

Fig. 21.1-2

If absolute data specification is programmed, the motion will be as follows:

```
N1 G31 G90 Z200
N2 X200 Z300
```

The block N1 starts a motion in the direction Z to the point with the coordinate of Z=200. If, after arriving of the external signal, the control stops at the point with the coordinate of Z=130, the displacement in the direction Z in the block N2 will be Z=300 - 130, i.e. Z=170.

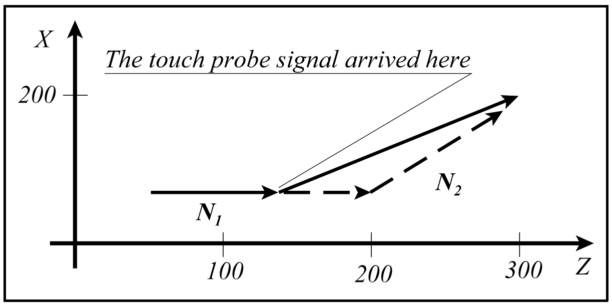


Fig. 21.1-3

21.2 Torque Limit Skip (G31)

The function G31 can also be used in the way by ***skipping the motion*** not by an external signal (for example by the signal of the probe) but when ***the torque of the motor at a specified axis reaches a given value***, for example, when the tool is pressed against a fixed surface. At the moment of reaching the torque limit or at the programmed endpoint of the block the control stops, and then it goes on to execute the next block.

It is the function

G31 P98 Qq v Ff

that realizes the actions mentioned above, where:

Q: the value of the programmable torque limit expressed in percent of the maximum torque of the motor. Q0 corresponds to 0% and Q255 corresponds to 100%. The values that can be given: Q1-Q254

v: the name and the endposition of the programmed axis on which the torque limit is monitored by the control. ***Only one axis address has to be specified***

F: the value of feed in mm/min or inch/min. (G94 state is needed).
During motion, the value of the *feed F* will be:
– the specified or modal value F if the bit #0 SKF of the parameter N3001 G31 Config is 0;
– the value F taken from the parameter N0311 G31 Feed if the bit #0 SKF of the parameter N3001 G31 Config is 1.

The command ***G31 P98 is not a modal one***; it is valid only in the block, in which it was programmed.

An example:

```
N1 G31 P98 Q50 Z30 F100
N2 G0 Z200
```

In the block N1, ***the tool reaches the workpiece surface at the point 'A'***. As from this point, the axis Z will not move already. Because the motor has not reached the torque limit Q50 (19.6%) yet, the control will not stop the motion.
The ***position error*** on the axis Z ***from the point 'A' to the point 'B'*** will increase continuously while the torque of the motor increases continuously.
Issuing the motion command continues up to ***the point 'B'*** where ***the motor reaches the torque limit***.

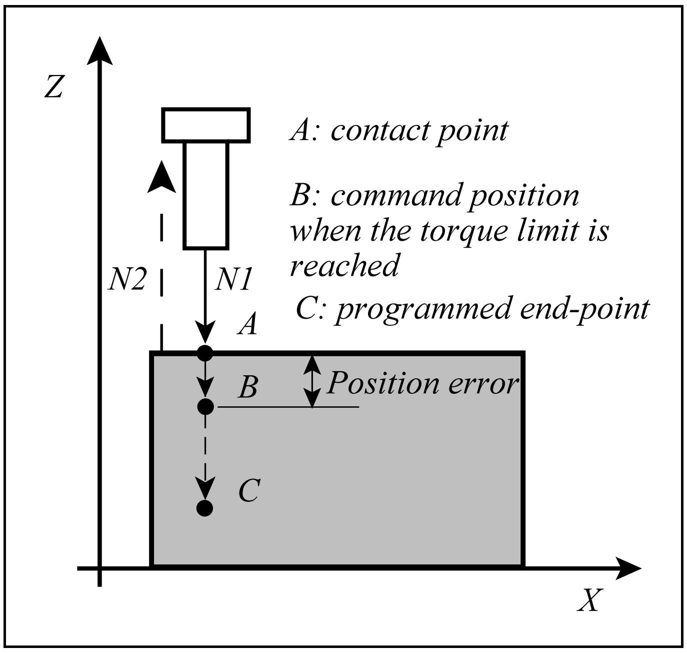


Fig. 21.2-1

At the point 'B', the control stops the motion, ***records the position of the point 'A', and then begins execution of the next block N2***.

The position error increases continuously increases from the point 'A' to the point 'B' while the torque of the motor increases because the tool pushes the workpiece. In order that the control does not run to a servo error during execution of the function, an error limit greater than the limit set in the parameter N0520 Serrl2 can be set in the parameter N3019 Servo Limit during Torque Limit Skip. The control will send the message '3157 Servo error during G31P98' only after exceeding this limit. This servo error does not cause emergency situation but it suspends execution of the program.

At the moment of reaching the torque limit, ***the position of the axis recorded at the point 'A' will be stored*** in the following ***macro variables***:
#5061, or #100151, or #_ABSKP[1]: the position of the axis 1;
#5062, or #100152, or #_ABSKP[2]: the position of the axis 2;
...
The position stored in the abovementioned macro variables will be:
– the position of the point 'A' if ***torque limit signal has been received***;
– the position of programmed endpoint of the block G31 P98 if ***torque limit signal has not been received***.

The position data will be stored
– always ***in the actual workpiece coordinate system***;
– ***without taking the actual length compensation*** (G43 and G44) into account.

Error messages
If the address Q is not filled in, the control will send the message '2004 The data Q is missing'.
If the value of the address Q is less than 1 or greater than 254, the control will send the message '2039 Q definition error'.
If more than one axis address is referred in the function, the control will send the message '2035 <axis address> axis definition is illegal', where <axis address> is the address of the programmed axis 2 according to the axis number order.
If the control does not receive torque (current) information about the programmed axis, it will send the message '2156 Probe status error on the channel 98'.

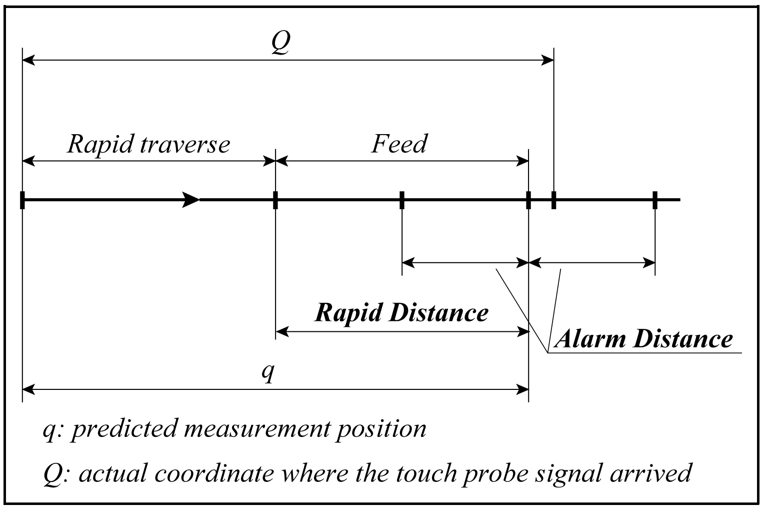
21.3 Automatic Tool Length Measurement (G36, G37)

Due to the command

G36 X_
G37 Z_

the length compensation of the tool changed will be measured and corrected along the axis X in the case of the G36, and along the Z in the case of the G37. The values of the X and Z are always interpreted as absolute data, in the coordinate system of the actual workpiece.

The motion is performed to the position ***q - Rapid Distance at the rapid traverse rate***, where Rapid Distance is the distance set in the parameter N3006 Rapid Distance X for the axis X, and in the parameter N3010 Rapid Distance Z for the axis Z; q is the value programmed at the addresses X, Z.


Q
Rapid traverse
Feed
Rapid Distance
Alarm Distance
q
q: predicted measurement position
Q: actual coordinate where the touch probe signal arrived
Fig. 21.3-1

Then, the motion continues ***at the feed rate*** until ***touch probe signal arrives or the control indicates error***. The error message '2104 Measurement position out of area' will be sent, if the touch probe signal arrives from a place which is out of the range with the radius of Alarm Distance, being around the predicted measurement position q (programmed at the address X or Z). The distance Alarm Distance is a value set in the parameter N3007 Alarm Distance X for the axis X, and in the parameter N3011 Alarm Distance Z for the axis Z.

In the block G36, G37, the value of the feed will be:
– a modal value from the program, if the bit #0 TLF of the parameter N3003 G36, G37 Config is 0; or
– a value set in the parameter N0312 G37 Feed Feed, if the bit #0 TLF of the parameter N3003 G36, G37 Config is 1.

If the measurement is completed successfully and the touch probe signal arrived at the point with the coordinate Q, the following compensation modifications will be possible in the length compensation registers called previously:
– ***geometry compensation*** will be modified if the bit #1 TMW of the parameter N3003 G36, G37 Config is 0;
– ***wear compensation*** will be modified if the bit #1 TMW of the parameter N3003 G36, G37 Config is 1.
Modifying the length compensation is as follows:
– ***the difference q-Q will be subtracted from the*** appropriate ***compensation*** if the bit #2 TCA of the parameter N3003 G36, G37 Config is 0;
– ***the difference q-Q will be added to the*** appropriate ***compensation*** if the bit #2 TCA of the parameter N3003 G36, G37 Config is 1.

Prior to starting the measurement, ***the appropriate length compensation has to be called***.
– The G36, G37 is one-shot command.
– The cycle G36, G37 is always executed in the coordinate system of the actual workpiece.
– The parameters Rapid Distance and Alarm Distance are always positive values. For these two parameters, the following condition has to be satisfied: Rapid Distance > Alarm Distance.
– The function can be called only in the state G15, G50, G50.1, G69, G94, otherwise an error message will be sent by the control.

Example

```
G55 G15 G50 G50.1 G69 G94
...
G0 X300 Z200
T505
X20 Z100
G37 Z50 F200
Z100
X200
Z50
G36 X40
X200
...
```

22 Safety Functions

The following three safety zones can be set on the control:
– ***Stroke ends:*** they determine the limit of axis travel, entering beyond which is forbidden. Switches or parameters are used for limitation.
– ***Working area limitation (Stored stroke check 2):*** it can be started and cancelled from program by the functions G22 and G23, respectively. It can be set by specification of parameters too. Both inside and outside forbidden areas can be defined.
– ***Area forbidden internally (Stored stroke check 3)***: an area moving into which is forbidden. It can be set using parameters.

22.1 Stroke End

The stroke ends limit axis travel. Stroke ends can be managed from switch or from parameter. They always forbid outside area.

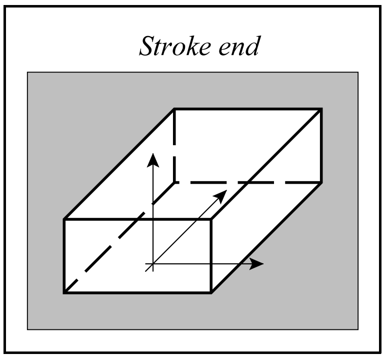


Fig. 22.1-1

Managing stroke end from switch

It is the PLC program that manages signals from the limit switches and transmits them to the control. When an axis moves onto a limit switch, the error message
'3018 Axis # on positive limit switch' or
'3019 Axis # on negative limit switch'
will be sent by the control, where # is the name of the axis.
A ***disadvantage*** of stoke limitation using limit switch is that ***the control begins to decelerate after moving onto the switch***. On the machines featured by high rapid traverse rate, ***it would be necessary to decrease the stroke to a great extent because of the long deceleration distance***, in order that the axis will be able to stop from rapid traverse. Furthermore, monitoring the stroke end prior to motion start does not work either.

Managing stroke end from parameter

In the case of managing stroke end from parameter, the control knows momentarily the distance to the stroke end, and it always begins to decelerate at the right moment in accordance with the axis speed. In this way, ***the machine stroke is fully utilized by parametric stroke end***.
When an axis moves onto a parametric stroke end, the error message
'3010 Positive limit on axis # obtained' or
'3011 Negative limit on axis # obtained'
will be sent by the control, where # is the name of the axis.
The parametric stroke ends are determined by the builder of the machine tool.
Monitoring the parametric stroke end will be effective only after positioning to the reference point. On those machines, where there is no absolute measuring system, i.e. reference point return is necessary, the rapid traverse rate is limited before positioning to the reference point, for safety reason. In the case of the axes equipped with absolute measuring system, the parametric stroke end is effective immediately after switching on.
Two stroke end ranges, A and B, can be set in parameter. During operation of the machine, ***the PLC program determines*** which stroke end range ***(A or B) should be effective*** on which axis and in which direction.

For example, in a normal case, the stroke end range A is effective on the axis Z. If, in the course of tool change, the change arm grips the tool, the PLC program will switch over to the stroke end range B; in this way, the stroke end range of the axis Z is adjusted within such narrow boundary that does not allow tearing the change arm off by the motion of the axis Z.

Leaving the stroke end

If, in the course of program run or manual moving, any of the axes moves onto a stroke end, ***leaving it will be possible by moving manually only***.

22.2 Working Area Limitation from Parameter/Program (G22, G23)

Working area limitation from parameter

Working area limitation can be given by setting the following parameters. Parameters can be set only in the Edit mode.

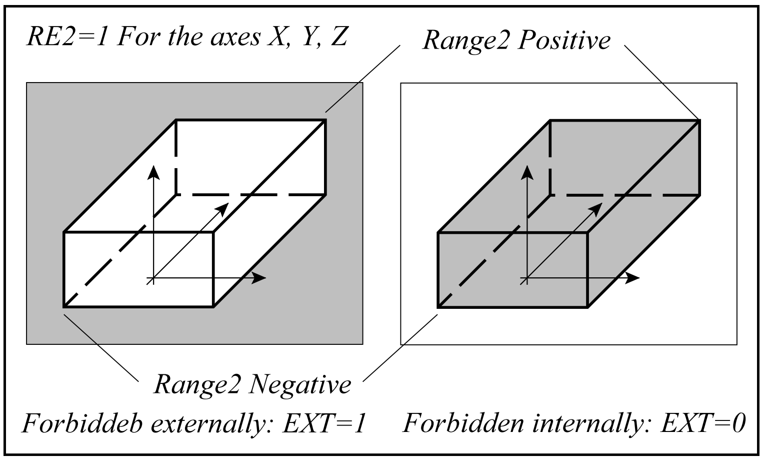

RE2=1 For the axes X, Y, Z
Range2 Positive
Range2 Negative
Forbiddeb externally: EXT=1
Forbidden internally: EXT=0

Fig. 22.2-1

With setting the parameter bit #1 RE2 of the parameter N1000 Range Enable to 1, it can be determined ***which axis*** will participate in working area limitation.

The point of positive and negative direction of the working area to be limited has to be given in the parameters N1006 Range2 Positive and N1007 Range2 Negative ***in the machine coordinate system***, for each axis.

The condition

N1006 Range2 Positive > N1007 Range2 Negative

has to be satisfied for all assigned axes!

Using the enable bit and the machine positions***, all the axes on the machine can be assigned in parameter*** for working area limitation.

It can be given at the bit #0 EXT of the parameter N1001 StrkContr ***whether the assigned working area will be forbidden internally*** (when EXT=0) ***or externally*** (when EXT=1).

The working area limitation has to be enabled with setting the bit #1 STE of the parameter N1001 StrkContr to 1. If the bit STE is 0, the working area limitation will not operate.

The bits EXT and STE has to be set for each channel. The working area limitation is always related to the axes belonging to the given channel.

Following the entering or rewriting the parameters, the single block

```
G23
```

has to be issued in one of the manual modes. As a result of this, the entered or modified parameters will be taken into account by the control.

The figure illustrates a working area limitation for the axes X, Y, Z. Certainly, either less or more axes can be assigned for this function.

If the machine moves into an area forbidden internally (EXT=0), the error message

‘3042 Internally forbidden area 2'

will be sent by the control.

If the machine moves into an area forbidden externally (EXT=1) in positive or negative direction, the error message

‘3040 Forbidden area 2 # +’ or

‘3041 Forbidden area 2 # -‘

will be sent by the control, where # is the name of the axis.

Leaving the area forbidden by parameter

If, in the course of program run or manual moving, any of the axes moves into forbidden area, the following method should be applied:

The case when the working area ***is forbidden externally.***

The area can be left ***by moving manually***, similarly to leaving the stroke end.

The case when the ***working area is forbidden internally***.

By switching the PLC flag on: The program has to be ***reset***, ***the error has to be deleted***. By switching the flag CP_LIM2DIS PLC on (e.g. ***pushing a button and keeping it pushed***), monitoring the forbidden area 2 can be suspended and leaving the area will be possible without rewriting the parameters. *For details the builder of the machine tool has to be asked* .

By parameter setting: Monitoring the working area limitation ***has to be cancelled by parameter setting STE=0***, the command G23 has to be issued in single block, ***the area has to be left by moving manually***, ***monitoring has to be started again*** by parameter setting STE=1, and the command G23 has to be issued again.

Working area limitation from program

The command

G22 X Y Z I J K P

starts monitoring the working area limitation. The motion ranges of the axes can be limited by this command. The meaning of the addresses of the command are as follows:

X: Limit along the axis X in the positive direction;
I: Limit along the axis X in the negative direction;
Y: Limit along the axis Y in the positive direction;
J: Limit along the axis Y in the negative direction;
Z: Limit along the axis Z in the positive direction;
K: Limit along the axis Z in the negative direction.

The following conditions have to be satisfied for data above:

$$X \geq I, \quad Y \geq J, \quad Z \geq K$$

All the coordinate data (X, Y, Z, I, J, K) ***have to be given in the machine coordinate system***.

It can be given at the address P whether moving beyond or into the assigned space is forbidden.

In the case of P=0, the internal zone of the assigned space is forbidden.

In the case of P=1, the external zone of the assigned space is forbidden.

Due to the command G22, the values of the range 2 set in parameter will be ignored by the control, and only those values will be taken into account that were given in the command G22.

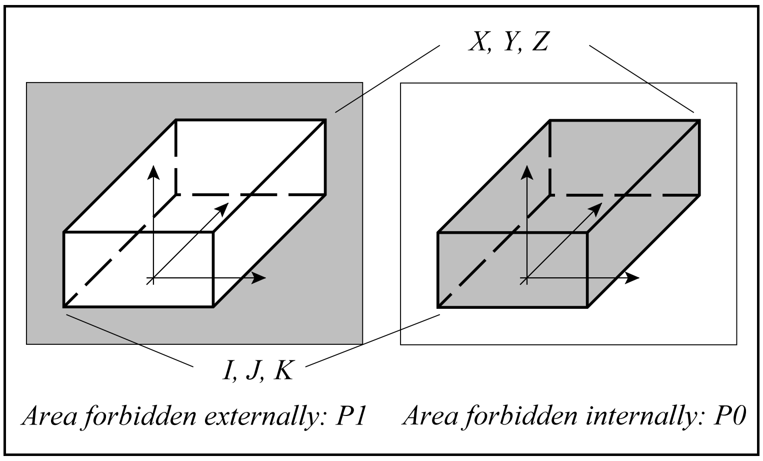


Fig. 22.2-2

The command

G23

cancels monitoring the working area limitation.

The command G23 deletes the borders of the range set in the command G22, and, at the same time, resets the values of monitoring the range 2 set in parameter.

– Working area limitation can be given only for ***main axes***.
– The commands G22 and G23 has to be given in independent block..
– Working area limitation will be effective after turning on and machine reference point return.
– In the case of X=I, Y=J, Z=K and P=0, the entire area is allowed.
– In the case of X=I, Y=J, Z=K and P=1, the entire area is forbidden.

If the machine moves into an area forbidden internally (G22 P0), the error message

‘3042 Internally forbidden area 2'

will be sent by the control.
If the machine moves into an area forbidden externally (G22 P1) in positive or negative direction, the error message

‘3040 Forbidden area 2 # +’ or

‘3041 Forbidden area 2 # -‘

will be sent by the control, where # is the name of the axis.

Leaving the forbidden area programmed using the function G22
If, in the course of program run or manual moving, any of the axes moves into forbidden area, the following method should be applied:

The case when the working area ***is forbidden externally.***

The area can be left ***by moving manually***, similarly to leaving the stroke end.

The case when the ***working area is forbidden internally***.

By switching the PLC flag on: The program has to be ***reset***, ***the error has to be deleted***. By switching the flag CP_LIM2DIS PLC on (e.g. ***pushing a button and keeping it pushed***), monitoring the forbidden area 2 can be suspended and leaving the area will be possible. *For details the builder of the machine tool has to be asked .*

From program: In manual mode, monitoring the working area limitation ***has to be cancelled using function G23***, ***the area has to be left by moving manually***, and ***monitoring has to be started again*** by issuing the complete command G22 again.

If, on the control, the forbidden area 2 is set from parameter, but working area limitation is issued using the command G22, the area specified by the G22 will be forbidden until it will be deleted by the G23. Then, the area 2 specified in parameter will be forbidden again.

22.3 The Area Forbidden Internally

An area always forbidden internally can be defined by parameters on the control. If one or maybe more of the axes moves into this area or to the border of it, the error message

‘3042 Internally forbidden area 3'

will be sent by the control.

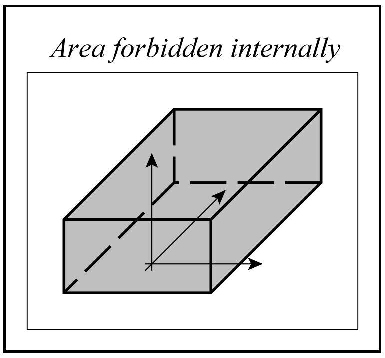


Fig. 22.3-1

Leaving the area forbidden internally

By switching the PLC flag on: The program has to be ***reset***, ***the error has to be deleted***. By switching the flag CP_LIM3DIS PLC on (e.g. ***pushing a button and keeping it pushed***), monitoring the area 3 forbidden internally can be suspended and leaving the area will be possible without rewriting the parameters. *For details the builder of the machine tool has to be asked.*

By parameter setting: If the machine moves into the abovementioned area forbidden internally, the bit #2 RE3 of the parameter N1000 Range Enable, i.e. enabling the monitoring the area forbidden internally, will have to be cancelled for each axis by writing the bits RE3 to 0. Then, the area has to be left by moving manually, and the bits #2 RE3 have to be set again.
The bits RE3 can be set only in the Edit mode.

22.4 Monitoring the Forbidden Area Prior to Motion Start

In the bit state #2 CBM=1 of the parameter N1001 StrkCont, ***prior to starting a motion command***, ***the control will check*** in automatic or manual data input mode or when a single block is being started, ***whether the endpoint of the given block will fall into one of the forbidden areas or not***.

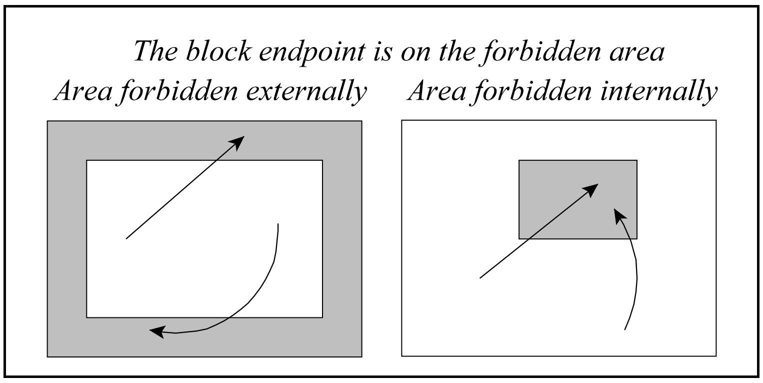

The block endpoint is on the forbidden area
Area forbidden externally
Area forbidden internally

Fig. 22.4-1

If the endpoint of the block ***falls beyond the stroke end within the forbidden area 1***,
the error message
‘2056 Endpoint on positive limit on axis #’
‘2057 Endpoint on negative limit on axis #’
will be sent by the control, where # is the name of the related axis; and ***motion will not be started***.
If the endpoint falls within ***the internally forbidden area 2***,
the error message
‘2060 Endpoint in internally forbidden area 2'
will be sent by the control, and ***motion will not be started***.
If the endpoint falls within ***the externally forbidden area 2***,
the error message
‘2058 Endpoint in forbidden area 2 # +’
‘2059 Endpoint in forbidden area 2 # -‘
will be sent by the control, where # is the name of the related axis; and ***motion will not be started***.
If the endpoint falls within ***the internally forbidden area 3***,
the error message
‘2061 Endpoint in internally forbidden area 3'
will be sent by the control, and ***motion will not be started***.

In the bit state #2 CBM=1 of the parameter N1001 StrkCont, ***prior to starting a motion command***, ***the control will check*** in automatic or manual data input mode or when a single block is being started, ***whether the path of the given block intersects the area forbidden internally***.

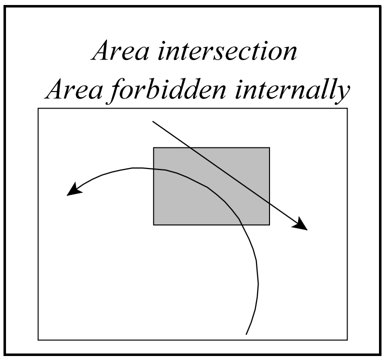


Fig. 22.4-2

If ***the path of the block intersects the internally forbidden area 2***, but the endpoint is not within the forbidden area,
the error message
‘2062 Entry into second internally forbidden area’
will be sent by the control, and ***motion will not be started***.
If ***the path of the block intersects the internally forbidden area 3***, but the endpoint is not within the forbidden area,
the error message
‘2063 Entry into third internally forbidden area’
will be sent by the control, and ***motion will not be started***.

The errors mentioned in the cases above can be eliminated by rewriting the programmed coordinates and by modifying the zero points or tool compensations.

23 Custom Macro

The conventional NC programming language describes the desired path and switches the various functions on or off by specifying codes G, M, S and T. Specific numerical value is assigned to a given address. For example, if the axes are to be moved to the position X50 Y100, the block

```
      G0 X50 Y100
```

will have to be programmed.

With the application of the macro language, ***it is not necessary to assign a specific numerical value***, e.g. X50, to a given address, but instead, ***the value of a variable can also be assigned to it***; for example, it can be written in the program that

```
      G#105 X#102 Y#110
```

where #105, #102 and #110 are the values of three different variables to which a value was assigned earlier.

In the programming language, various ***arithmetical expressions*** and ***functions*** can be used, for example addition, square-root extraction, sine function etc.

Assignment commands, ***condition check*** commands, ***branch*** commands and commands executing ***cycle*** can be used.

The programming language enables such subprograms, ***macros to be called*** to which ***arguments*** (parameters) ***can be passed*** from the calling block.

With parameter specification, such so-called ***system macros or system subprograms*** can be generated that can be used by the user to extend or modify the commands of conventional programming language of codes G in accordance with his demands.

23.1 Variables of the Programming Language

In the main program, subprograms and macros, ***variables can also be assigned*** to the addresses, instead of specific numerical values.
Value within a permissible range ***can be assigned to the variables***. By the use of variables, programming can be made much flexible.
Variables can be classified as

local variables that are used for passing argument in macro calls;
common variables that can be accessed at every level of macro call; and
system variables.

The system variables are those internal data of the control that can be read out or rewritten from the part program.

23.1.1 Referring to Variables

Variables can be referred by ***number***, but system variables can be referred either by number or by ***symbol***.
Referring to a variable has to always be begun with the number sign #.

<u>Referring to variable by number</u>

The identifier of the variable is ***a number following the number sign #***:

#<number>

For example:

#12
#138
#5106

A variable ***can also be referred indirectly***, by a formula: ***#[<formula>]***
For example:

#[#120] means that the variable No. 120 contains the serial number of the variable referred to.
#[#120-4] means that subtraction 4 from the number contained in the variable No. 120 gives the number of the variable referred to.

<u>Referring to system variables by symbol</u>

Symbolical reference is also begun with the number sign # followed by the character _ (***underscore symbol***):

#_<symbol>

For example:

#_ALM (error message)

In certain cases, an ***index*** has to also be added to the symbolic variables. The index has to be put into ***brackets [ ]***:

#_<symbol>[index]

For example:

#_ABSIO[3] means block end position of the axis 3, axis index: [3].

In the words of the program block, the various addresses can take on not only numerical values, but also values of variables. In the case of referring to variable behind addresses, the minus sign (−) or the operator I can also be used wherever it is permitted in the case of numerical values. For example:

G#102

if #102=1.0, this reference is equivalent to G1

XI−#24

if #24=135.342, this reference is equivalent to XI−135.342

– It is not permitted to refer to a variable behind addresses of block number N and conditional block /. The first address N written in the block is considered to be block number by the control.

– The number of a variable cannot be substituted by a variable, i.e. writing ##120 is not permitted. The correct specification is #[#120].

– If a variable is used behind an address, the value of the variable will not have to exceed the range of values permitted for the given address. For example, after the value specification

#112=123456789

the reference

M#112

will cause error message.

– If a variable is used behind an address, the value of the variable will be rounded to the significant digit corresponding to the address. For example:

if the #112=3.23,	the M#112 will be M3,
if the #112=3.6,	the M#112 will be M4.

23.1.2 Number Representation of Macro Variables

Disregarding few exceptions, macro variables are floating-point numbers. The macro variables being not floating-point ones will have special marking in their description.

In the control, representation of the floating-point numbers follows the ***double precision representation of the floating-point numbers*** in accordance with ***the standard IEEE 754***. These numbers are represented on 64 bits.

Using the double precision representation of the floating-point numbers, the numbers

from $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$

and the 0 can be represented with precision of 15-16 digits. The ***decimal point*** (.) has to be used during input, but it is not necessary when the numbers to be input are integer numbers:

#100=256

Neither leading zeros nor following zeros have to be given. The positive sign (+) can be omitted:

#100=134.89654

23.1.3 Local Variables: #1 – #33

The local variables are used by the macro program at a given point, locally.

Generally, the local variables are used ***to pass arguments***.

The local variables are ***multi-level*** variables; different levels belong to the main program and to the various macro calls, and that is because they are called local. For example, the value of the #1 can be different in the main program than, let us say, in the level 2 of the macro calls.

After returning from macro call, the local variables of the given level will be eliminated to #0, they will be deleted to vacant. ***The local variables of the main program will be deleted at the end of the program***.

Fitting the addresses of the arguments and the local variables, and managing the levels are covered by the section 23.3 **Calling the Macros, System Macros and System Subprograms** on page 336.

The local variables the address of which is not included in the argument assignment is null and can be used freely.

23.1.4 Common Variables: #100 - #499, #500 - #999

Unlike the local variables, ***the common variables are identical in the whole channel*** regardless of whether they are used in the main program, subprogram or macro and in which level of the macro call.
In the system, ***the use of common variables absolutely free***, they do not have any dedicated role.

The following two groups of the common variables are distinguished:

The common variables from #100 to #499 that will be deleted upon power-off.
The common variables from #500 to #599 the value of which will be retained even after power-off.

The common variables from #500 to #999 can be made write-protected using the parameters N1702 Write Prt Low and N1703 Write Prt Hig.
The first element of the array to be protected has to be written in the parameter N1702 Write Prt Low, but the last element of the array declared protected has to be written in the parameter N1703 Write Prt Hig.
For example, if the common variables from#530 to #540 are to be made write-protected, the parameter settings N1702 Write Prt Low=530 and N1703 Write Prt Hig=540 will have to be applied.

If the control manages several channels, ***an array of the common variables can be made accessible in each channel*** using parameter.
The parameter N1700 No. of Common #100 determines the number of the macro variables from #100 to #499 that can be called from each channel. In each channel, the macro variables with the number from 100 to the parameter 100 + No. of Common #100 will be common. This parameter has to be smaller than 400.
If the value of the parameter, let us say, 40, the macro variables from #100 to #139 will be common for each channel.
The parameter N1701 No. Of Common #500 determines the number of the macro variables from #500 to #999 that can be called from each channel. In each channel, the macro variables with the number from 500 to the parameter 500 + No. of Common #500 will be common. This parameter has to be smaller than 500.
If the value of the parameter, let us say, 30, the macro variables from #500 to #529 will be common for each channel.

23.1.5 Notation Used in Description of System Variables

The system variables are those internal data of the control that can be read out or rewritten from the part program.

The axes can be identified only from 1 to 20 on the macro variables belonging to the axes and identified by a number under 10000. For the case of greater axis number, the identification numbers over 100000 has been introduced. Referring to axes can be done on the numbers from 1 to 50. For example:

```
#100001
```

can also be used to identify the block end position of the axis 1. Certainly, referring to the data above can also be done using numbers smaller than 10000, and symbols.

Notation used in description of system variables is as follows:

[n]: the index of the variable. It can be, for example the number of an axis or a spindle;
R: an attribute of the variable: read-only variable;
W: an attribute of the variable: write-only variable;
R/W: an attribute of the variable: readable and writable variable.

23.1.6 Vacant Variable. Constants

Number	Symbol	Attribute	Description
#0, #3100	#_EMPTY	R	Empty constant
#3101	#_PI	R	π= 3.14159...
#3102	#_E	R	Base of natural logarithm: e=2.71828...

Vacant variable #0, #3100, #_EMPTY (R)
When a macro is called, if value is not assigned to an address, the value of the local variable belonging to that address will be vacant in the body of the macro.
For example: After calling

G65 P100 X20 Y30,

the value of the local variable #1 in the macro O0100 will be vacant because value was not assigned to the address A in the call G65. It can be decided with the checking

#1 EQ #0

in the body of the macro whether the address A was filled in the course of calling the macro or not.
The vacant variable and the number 0 are not identical ones, they differ from each other!

The difference between the effects of the vacant variable and a variable having the value 0 is as follows:
Referring to the ***vacant*** variable in address:

```
If #1=<vacant>          If #1=0

G90 X20 Y#1          G90 X20 Y#1
        |                    |
  G90 X20            G90 X20 Y0
```

Vacant variable in *assignment* command:

```
            If #1=<vacant>            If #1=0

               #2=#1                 #2=#1
                 |                     |
             #2=<vacant>                #2=0

              #2=#1*3               #2=#1*3
                 |                     |
               #2=0                  #2=0

              #2=#1+#1              #2=#1+#1
                 |                     |
               #2=0                  #2=0
```

The difference between ***the vacant variable*** and ***a variable having the value 0*** in the case of *condition checking*:

```
            If #1=<vacant>            If #1=0

             #1 EQ #0                #1 EQ #0
                 |                     |
             satisfied           not satisfied

             #1 NE 0                 #1 NE 0
                 |                     |
             satisfied           not satisfied

             #1 GE #0                #1 GE #0
                 |                     |
             satisfied           not satisfied

             #1 GT 0                 #1 GT 0
                 |                     |
             satisfied           not satisfied
```

23.1.7 Variables Between the Part Program and the PLC Program

Information exchange between the part program and the PLC program can be realized by the variables described below.

☞ ***Warning!*** *The kind of information the PLC program transmits to the part program and receives from the part program through various system variables is determined the builder of the machine tool.*

Number	Symbol	Attribute	Description
#1000...#1031	#_UI[n] n=0 - 31	R	32 ***bit variables*** of the PLC program transmitted to the control. Value set: 0, 1
#1032	#_UIL[n] n=0	R	32 ***bit variables*** of the PLC program transmitted to the control as a ***32-bit integer number***. Value set: 0 ... 2^{32}-1

Number	Symbol	Attribute	Description
#1033...#1035	#_UIL[n] n=1, 2, 3	R	3 ***floating-point variables*** of the PLC program transmitted to the control.
#1100...#1131	#_UO[n] n=0 - 31	R/W	32 ***bit variables*** of the part program transmitted to the PLC. Value set: 0, 1
#1132	#_UOL[n] n=0	R/W	32 ***bit variables*** of the part program transmitted to the PLC as a ***32-bit integer number***. Value set: 0 ... 2^{32}-1
#1133...#1135	#_UOL[n] n=1, 2, 3	R/W	3 ***floating-point variables*** of the part program transmitted to the PLC.

23.1.8 Messages of the Part Program

From the macro program, errors can be indicated and messages can be sent to the operator:

Number	Symbol	Attribute	Description
#3000	#_ALM	W	Error message; it can be deleted by the button Cancel
#3006	#_MSGSTP	W	Stop accompanied by a message; continuation by Start
#3106	#_MSG	W	Displaying the message in the program list window
#3107	#_MSGBOX	R/W	Sending the message to the Windows window

Error message: #3000, #_ALM (W)
With the value assignment

#3000=nnn(ERROR INDICATION)

or

#_ALM=nnn(ERROR INDICATION),

numbered (nnn: maximum three digits) and/or text error message can be sent. The text has to be written between round brackets (,).
If an error in the macro is detected by the program, i.e. the program runs on a branch where value was assigned to the variable #3000, the program will be executed up to the previous block, and then the execution will be suspended, and the error message given between brackets or the code of the message will be displayed on the screen in the form of

ii4nnn00 (ii: the number of the channel in which the error occurred),

i.e. 4000 will be added to the number nnn given at the value #3000. If a number is not given, the code of the message will be 4000; if a text is not given, only the code will appear. The error message can be deleted by the button CANCEL.

Stop with a message: #3006, #_MSGSTP (W)
With the value assignment

#3006=nnn(MESSAGE)

or

    #_MSGSTP=nnn(MESSAGE),

execution of the program will stop, and the message given between brackets or the code of the message will be displayed on the screen in the form of

ii5nnn00 (ii: the number of the channel in which the error occurred)

i.e. 5000 will be added to the number nnn given at the value #3006. If a number is not given, the code of the message will be 5000; if a text is not given, only the code will appear. Pushing the button START the execution of the program will continue, and the message will disappear from the screen. This command is useful in the cases when operator intervention is needed during execution of the command.

Displaying a message in the program list window: #3106, #_MSG (W)

With the value assignment

    #3106=nnn(MESSAGE)

or

    #_MSG=nnn(MESSAGE)

the program continues without stop, and ***the text of the message will be displayed in the top line of the Program list window*** in the form of:

MSGnnn: (MESSAGE)

The text of the message remains there until it is overwritten by a new #3106 or #_MSG instruction. RESET, program end (M30) deletes the message.

In order that the message text appears on the top line of the program list, **the displaying has to be enabled** using the function keys as follows:

F5 View - F1 Program list - F9 Settings - F3 #_MSG info

It can be used for indication of program parts indication for example:

```
...
#3106=1(roughing by the use of a diam. 30 milling cutter)
...
...
#_MSG=2(finishing by the use of a diam. 20 milling cutter)
...
```

Sending a message to the Windows window: #3107, #_MSGBOX (R/W)

With the value assignment

    #3107=nnn(MESSAGE)

or

    #_MSGBOX=nnn(MESSAGE)

execution of the program will stop (there will be STOP status), and the message given between brackets and the code of the message will be displayed in the message row of the control in the form of

ii6nnnjj (ii: the number of the channel in which the message was issued).

The message will appear not only in the upper status line but in the centre of the screen, **in a Windows message window** too.

It is the code of the message which determines the type of the message window. If the message assignment is

between **nnn=100 - 199**, the message window will appear **with an 'OK' button**, so the operator can only accept the displayed message. If the message assignment is between **nnn=200 - 299**, the message window will appear **with a 'Yes' button and a 'No' button**. In this case, the operator has alternative. In case of any other value assignment, an error message will be sent by the control.

After responding to the messages appeared in the Windows window, the message will be deleted and the #3107 variable will take the following values in the different cases:

#3107=0: the **X (closing)** button was clicked or the **CANCEL** button was pushed;
#3107=1: the **'OK'** button was clicked;
#3107=2: the **'Yes'** button was clicked;
#3107=3: the **'No'** button was clicked.

During displaying the message, if the intension is to display a given part of the text in several rows in the message window, it is possible to divide up the text by the use of the '\n' character and insert a line character. In the case of displaying a value, the macro variable formatting described at the instruction DPRNT can be applied.

Example 1:
```
##3107=100(Values of the measurement results\nX length:
     #140[53]\nY length: #141[53])
```

Example 2:
```
#140=0.3458
#141=0.9123
(Measurement)
(Displaying the result of the compensation calculation)
#_MSGBOX=200(Values of the tool wear\nX diameter wear:
     #140[53]\nZ-direction wear: #141[53]\nDo you want to
     input the new compensation?)
IF [[#_MSGBOX] EQ 2] GOTO10
GOTO20
N10 (Yes branch)
(Inputting the wear)
N20
(Continuing the program)
```

23.1.9 Clock, Timers and Part Counters

Number	Symbol	Attribute	Description
#3001	#_CLOCK1	R/W	Millisecond timer, usable freely
#3002	#_CLOCK2	R/W	Cutting time (ms)
#3011	#_DATE	R	Date: year/month/day
#3012	#_TIME	R	Time: hour/minute/second
#3901	#_PRTSA	R/W	Number of the parts produced
#3902	#_PRTSN	R/W	Number of the parts to be produced

Millisecond timer: #3001, #_CLOCK1 (R/W)
The value of this variable is writable and readable.

The time elapsed between two points in time that can be measured in millisecond. Measuring the value of the variable is started from zero at the moment of turning the control on, and performed up. Time is always measured when the control is on. Using the command

```
        #3001=0
```

or

```
        #_CLOCK1=0,
```

the variable can be set to zero by value assignment from the program; in this case measurement starts from zero, and later the value can be queried from the program, using the command

```
        #100=#3001
```

or

```
        #100=#_CLOCK1.
```

Cutting time: #3002, #_CLOCK2 (R/W)
The value of this variable is writable and readable. The time elapsed during machining with feed in automatic mode and start state (G1, G2 etc.) is measured in millisecond, cumulatively from the beginning of the control life. The value of the variable can be read on the display Cutting time of the screen Clock times and part counters.

Date: #3011, #_DATE (R)
The actual date can be read from the variable in the format year/month/day.
After using the command

```
        #100=#3011
```

or

```
        #100=#_DATE,
```

if the value of the variable #100 is, for example,

```
        20140518,
```

it will mean that the date is: 2014 (year), 05 (month, May), 18 (day, 18).

Time: #3012, #_TIME (R)
The actual time can be read from the variable in the format hour/minute/second.
After using the command

```
        #100=#3012
```

or

```
        #100=#_TIME,
```

if the value of the variable #100 is, for example,

```
        20140518,
```

it will mean that the time is: 15 hours (3 hours p.m.), 32 minutes, 41 seconds.

Number of the parts produced/to be produced: #3901, #_PRTSA / #3902, #_PRTSN (R/W)
The values of these variables are writable and readable. The number of the parts produced is accumulated by the control in the counter #_PRTSA having the number #3901. The control will increase the content of the counter by 1 in the course of execution of each function M02, M03 or function M given in the parameter N2305 Part Count M.
When the number of the parts produced reaches the number of the parts to be produced (the counter #_PRTSN having the number #3902) the PLC will be notified.

The number of the parts produced	#3901, #_PRTSA
The number of the parts to be produced	#3902, #_PRTSN

The value of the numbers of the parts produced and to be produced can be read on the display Produced and to be produced of the screen Clock times and part counters.

23.1.10 Variables Influencing the Operation of the Automatic Mode

Number	Symbol	Attribute	Description
#3003	#_CNTL1	R/W	Control variable 1 (block-by-block)
#3003 bit 0	#_M_SBK	R/W	Disabling the block-by-block execution. Value set: 0, 1
#3004	#_CNTL2	R/W	Control variable 2
#3004 bit 0	#_M_FHD	R/W	Disabling the Stop. Value set: 0, 1
#3004 bit 1	#_M_OV	R/W	Disabling the Override and the Stop. Value set: 0, 1
#3004 bit 2	#_M_EST	R/W	Disabling the exact Stop. Value set: 0, 1

☞ ***Warning!*** *By reset and block end, the values of the bits set at the variables will be deleted!*

Control variable 1: #3003, #_CNTL1 (R/W)
If the value of the variable #3003 or #_CNTL1 is 1 (or an uneven number), the control, in the state of the block-by-block execution, will not stop after execution of a block until the value of this variable will be 0.
With writing the variable
#_M_SBK ,
disabling the block-by-block execution can also be referred using bits (assigning 0 or 1).
By the power on or reset the value of the variable will be 0.
The block-by-block execution
will not be disabled if the value of the variable is 0;
will be disabled if the value of the variable is 1.

Control variable 2: #3004, #_CNTL2 (R/W)
The following values cab be assigned to the variable #3004 or #_CNTL2:

Value of the #3004 or the #_CNTL2	**Disabling the Exact stop G61,G9**	**Disabling the Override and the Stop**	**Disabling the Stop**
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

The Exact stop, the Override and the Stop
will not be disabled if the number written in the right column is 0;
will be disabled if the number written in the right column is 1.
The suppressions above can also be specified by assigning 0 or 1 to the bit-type variables:
The case of the variable **#_M_FHD** (disabling the Stop):
0: there is no disabling;
1: there is disabling.
The block-by-block execution is not disabled.
The case of the variable **#_M_OV** (disabling the Override and the Stop) (the state G63):
0: there is no disabling;
1: there is disabling.
The block-by-block execution is also disabled.
The case of the variable **#_M_EST** (disabling the Exact stop G61, G9) (the state G64):
0: there is no disabling;
1: there is disabling.

23.1.11 Querying the Block Search and Test Statuses

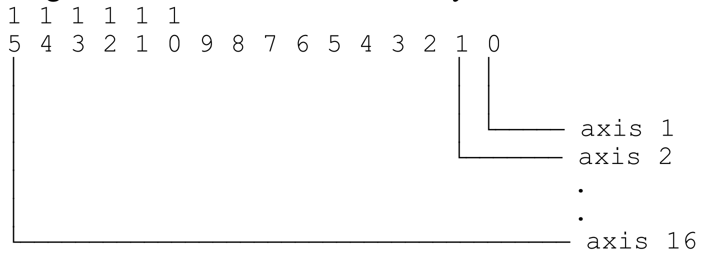
Number	Symbol	Feature	Description
	#_CNTBS	R	Querying the block search and test statuses
bit 0	#_BSEARCH	R	Block search status
bit 1	#_TEST	R	Test mode
bit 2	#_MCHLOCK	R	Machine locked mode

Querying the block search and test statuses: #_CNTBS (R)
All three of statuses in one can be queried at the variable #_CNTBS.
Bit-type querying is also possible for the variable #_CNTBS:
#_BSEARCH: The control executes block search:
=0: it does not execute block search;
=1: it executes block search.
#_TEST: Test mode:
=0: the control is not in test mode;
=1: the control is in test mode.
#_MCHLOCK: The machine is locked:
=0: the machine is not locked;
=1: the machine is locked.

23.1.12 Status of Mirror Image

Number	Symbol	Feature	Description
#3007	#_MIRIMG	R	Status of Mirror Image

By the reading of the variable #3007 it can be determined on which axis a valid mirroring command is entered. The variable is only readable.
Interpreting the value of the variable binary:



The meaning of the bits is as follows:
0: there is no mirroring;
1: mirroring is turned on.
For example, if the value of the variable is 5, it means that mirroring is turned on at the axes 1 and 3. The axis number means physical axis number; it is determined by parameter which axis of what name belongs to which physical axis number.

23.1.13 Number of the Main Program

Number	Symbol	Attribute	Description
#4000	#_MAINO	R	The number of the main program being executed

The number of the actual main program being executed will always be indicated even if a subprogram is in progress. If the automatic mode is interrupted and a program is executed in the MDI mode, the number of the program being executed in the MDI mode will be indicated. If the name of the main program begins with the letter other than O and contains more than 8 digits, the value of the #4000 will be 0.

23.1.14 Modal Information

The modal information valid in the previous block can be obtained by reading the system variables from #4001 to #4199.

Number	Symbol	Attribute	Description
#4001...#4039	#_BUFG[n] n=1-39	R	For each group, the code of the modal function G given last. n: the number of the group of codes G
#4101	#_BUFA	R	The value of the auxiliary function given last and the address of which was defined in the parameter N1333 Aux Fu Addr1.
#4102	#_BUFB	R	The value of the auxiliary function given last and the address of which was defined in the parameter N1334 Aux Fu Addr2.
#4103	#_BUFC	R	The value of the auxiliary function given last and the address of which was defined in the parameter N1334 Aux Fu Addr3.
#4107	#_BUFD	R	The value of the address D given last.
#4108	#_BUFE	R	The value of the address E given last.
#4109	#_BUFF	R	The value of the address F given last.
#4111	#_BUFH	R	The value of the address H given last.
#4113	#_BUFM	R	The value of the address M given last.
#4114	#_BUFN	R	The value of the block number N given last.
#4115	#_BUFO	R	The value of the program number O given last.
#4119	#_BUFS	R	The value of the address S given last.
#4120	#_BUFT	R	The value of the address T given last.
#4130	#_BUFWZP	R	The number of the additional coordinate system given last (G54.1 Pn)

The modal information valid in the block being executed just at the moment can be obtained by reading the variables from #4201 to #4399.

Number	Symbol	Attribute	Description
#4201...#4239	#_ACTG[n] n=1-39	R	For each group, the code of the modal function G valid in the block being executed. n: the number of the group of codes G
#4301	#_ACTA	R	The value of the auxiliary function valid in the block being executed and the address of which was defined in the parameter N1333 Aux Fu Addr1.
#4302	#_ACTB	R	The value of the auxiliary function valid in the block being executed and the address of which was defined in the parameter N1334 Aux Fu Addr2.
#4303	#_ACTC	R	The value of the auxiliary function valid in the block being executed and the address of which was defined in the parameter N1334 Aux Fu Addr3.
#4307	#_ACTD	R	The value of the address D valid in the block being executed.
#4308	#_ACTE	R	The value of the address E valid in the block being executed.
#4309	#_ACTF	R	The value of the address F valid in the block being executed.
#4311	#_ACTH	R	The value of the address H valid in the block being executed.
#4313	#_ACTM	R	The value of the address M valid in the block being executed.
#4314	#_ACTN	R	The value of the block number N valid in the block being executed.
#4315	#_ACTO	R	The value of the program number O valid in the block being executed.
#4319	#_ACTS	R	The value of the address S valid in the block being executed.
#4320	#_ACTT	R	The value of the address T valid in the block being executed.
#4330	#_ACTWZP	R	The number of the additional coordinate system valid in the block being executed (G54.1 Pn)

The modal information valid in the interrupted block at the moment of calling the interruption macro can be obtained by reading the variables from #4401 to #4599.

Number	Symbol	Attribute	Description
#4401...#4439	#_INTG[n] n=1-39	R	For each group, the code of the modal function G valid in the interrupted block. n: the number of the group of codes G
#4501	#_INTA	R	The value of the auxiliary function valid in the interrupted block and the address of which was defined in the parameter N1333 Aux Fu Addr1.
#4502	#_INTB	R	The value of the auxiliary function valid in the interrupted block and the address of which was defined in the parameter N1334 Aux Fu Addr2.
#4503	#_INTC	R	The value of the auxiliary function valid in the interrupted block and the address of which was defined in the parameter N1334 Aux Fu Addr3.
#4507	#_INTD	R	The value of the address D valid in the interrupted block.
#4508	#_INTE	R	The value of the address E valid in the interrupted block.
#4509	#_INTF	R	The value of the address F valid in the interrupted block.
#4511	#_INTH	R	The value of the address H valid in the interrupted block.
#4513	#_INTM	R	The value of the address M valid in the interrupted block.
#4514	#_INTN	R	The value of the block number N valid in the interrupted block.
#4515	#_INTO	R	The value of the program number O valid in the interrupted block.
#4519	#_INTS	R	The value of the address S valid in the interrupted block.
#4520	#_INTT	R	The value of the address T valid in the interrupted block.

Number	Symbol	Attribute	Description
#4530	#_INTWZP	R	The number of the additional coordinate system valid in the interrupted block (G54.1 Pn).

☞ ***Warning!*** *In the case of modal codes G containing dot, for example G51.1, the code will be given back by the control in such a way so that the number standing after the dot will be written before the dot. Due to the command*

```
        #100=#4022,
```

the value of the ***#100 will be 151 in the state G51.1, and it will be 150 in the state G50.***

The following table illustrates ***sorting the modal codes G by group number***.
The one-shot and non-modal codes G belong to the group 0. The group 0 cannot be queried, so the codes G belonging to it are not shown in the table.

Group number	The modal codes G belonging to the group	Function
1	G0, G1, G2, G3, G33	Interpolation
2	G17, G18, G19	Plane selection
3	G90, G91	Absolute/incremental programing
4	G22, G23	Forbidden area on/off
5	G94, G95	Feed per minute/revolution
6	G20, G21	Inch/metric data specification
7	G40, G41, G42	Radius compensation off/on
8	G43, G44, G43.7, G49	Length compensation on/off
9	G73, G74, G76, G80, G81, G82, G83, G84, G84.2, G84.3, G85, G86, G87, G88, G89	Drilling cycles
10	G98, G99	Returning from drilling cycle to the initial/R point
11	G50, G51	Scaling off/on
12	G66, G66.1, G67	Modal macro call on/off
13	G96, G97	Constant surface speed on/off
14	G54, G54.1, G54.2, G55, G56, G57, G58, G59	Selecting the workpiece coordinate system
15	G61, G62, G63, G64	Feed control functions

Group number	**The modal codes G belonging to the group**	**Function**
16	G68, G69	In-plane rotation on/off
17	G15, G16	Polar coordinate off/on
18		
19		
20		
21	G12.1, G13.1	Polar interpolation on/off
22	G50.1, G51.1	Mirroring off/on
23		
24		
25		
26		
27		
28		
29		
30		
31	G50.2, G51.2	Polygonal turning off/on
32		
33		
34	G80.8, G81.8	Electronic gear box off/on
35		
36		
37		
38		
39		

23.1.15 Position Information

Number	Symbol	Attribute	Description
#5001...#5020	#_ABSIO[n] n=1-50	R	The block end position of the axis 1 ... n.
#100001...#100050			
#5021...#5040	#_ABSMT[n] n=1-50	R	The position of the axis 1 ... n in the machine coordinate system.
#100051...#100100			
#5041...#5060	#_ABSOT[n] n=1-50	R	The position of the axis 1 ... n in the workpiece coordinate system.
#100101...#100150			
#5061...#5080	#_ABSKP[n] n=1-50	R	The skip position of the axis 1 ... n in the workpiece coordinate system.
#100151...#100200			

☞ ***Warning!*** The read-out of the position information is done in accordance with the axis number. Prior to using this manual, you have to ask the builder of the machine tool for information about the axis numbers.

For example, let us have a two-channel control. There are axes X, Y and Z in the first channel, and axes X and Z in the second channel. Assignment of the axes by the builder of the machine tool is as follows:

The channel 1:

X – the axis 1
Y – the axis 2
Z – the axis 3

The channel 2:

X – the axis 4
Z – the axis 5

If the position information is to be queried in the channel 1, the index 3 will have to be used in the case of the axis Z; but the index 5 will have to be used if query is to be done in the channel 2.

Block end position of the axes: #5001...#5020, #100001...#100050, #_ABSIO[n] (R)
The block end position of the axes will be given back
in the coordinate system of the actual workpiece (G54, G55, ...);
using orthogonal coordinates;
disregarding all the compensations (length, radius).

Machine position of the axes: #5021...#5040, #100051...#100100, #_ABSMT[n] (R)
The machine position of the axes will be given back
in the coordinate system of the machine (G53). If the given axis moves, the value given back will change continuously.

Workpiece position of the axes: #5041...#5060, #100101...#100150, #_ABSOT[n] (R)
The workpiece position of the axes will be given back

in the coordinate system of the actual workpiece (G54, G55, ...);
using orthogonal coordinates;
having regard to all the compensations (length, radius). If the given axis moves, the value given back will change continuously.

Skip position: #5061...#5080, #100151...#100200, #_ABSKP[n] (R)
The position in the block G31 where the touch probe signal is received. The position will be given back
in the coordinate system of the actual workpiece (G54, G55, ...);
using orthogonal coordinates;
having regard to all the compensations (length, radius).
If the touch probe signal is not received, the variables above will take on the end point position programmed in the block G31.
After receiving the touch probe signal, de axis will decelerate and stop. The end point position of the block G31 #_ABSIO(n) will on the value being valid after deceleration and stop.

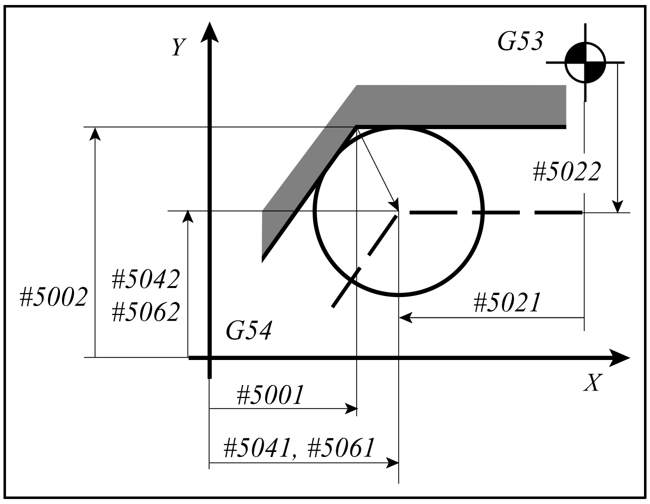


Fig. 23.1.15-1

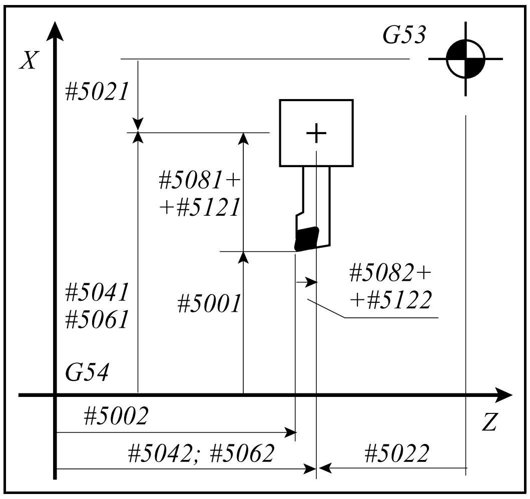


Fig. 23.1.15-2

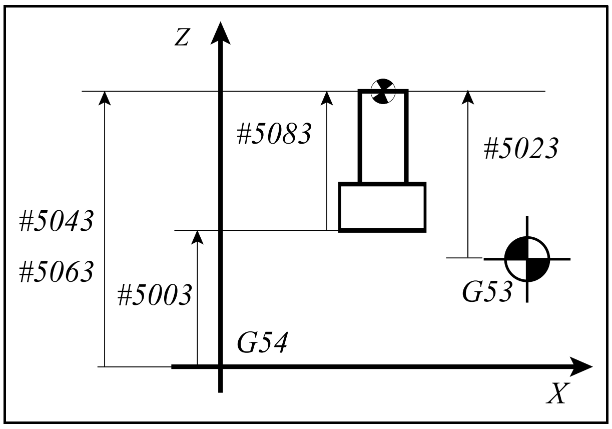


Fig. 23.1.15-3

23.1.16 Value of the Actual Length Compensation

Number	Symbol	Attribute	Description
#5081 #100201	#_TOFSWX	R	The value of wear compensation taken into account at the axis X.
#5082 #100202	#_TOFSWZ	R	The value of wear compensation taken into account at the axis Z.
#5083 #100203	#_TOFSWY	R	The value of wear compensation taken into account at the axis Y.
#5121 #100901	#_TOFSGX	R	The value of geometry length compensation taken into account at the axis X.
#5122 #100902	#_TOFSGZ	R	The value of geometry length compensation taken into account at the axis Z.
#5123 #100903	#_TOFSGY	R	The value of geometry length compensation taken into account at the axis Y.

From the variables above, the value of length compensation (geometry+wear) taken into account in the block being executed can be read out, for the axes X, Z and Y.

23.1.17 Other Position Information

Number	Symbol	Attribute	Description
#5101...#5120 #100251...#100300	#_SVERR[n] n=1-50	R	Lag of the axis 1 ... n.
#100651...#100700	#_MIRTP[n] n=1-50	R	Position offset made by handwheel at the axis 1 ... n.
#5181...#5200 #100801...#100850	#_DIST[n] n=1-50	R	The value of the distance to go at the axis 1 ... n.

Lag of the axes: #5101...#5120, #100251...#100300, #_SVERR[n] (R)
From the variables, the lag of the axis servo loop (servo-follower error) can be read out, in input unit of measurement.

Position offset made by handwheel: #100651...#100700, #_MIRTP[n] (R)
If, in automatic mode, even in the course of motion, position (zero point) of the axes are being corrected using handwheel, the extent of correction can be read out from the variables above

for each axis, in input unit of measurement.

The value of the distance to go: #5181...#5200, #100801...#100850, #_DIST[n] (R)
From the variables, the values of the distance to go, i.e. the distance to go to the end position in the block being executed, can be read out for each axis, in input unit of measurement. It is the value the control displays on the screen of the position as distance to go.

23.1.18 Values of the Tool Compensation Memory

Number	Symbol	Attribute	Description
#10001...#10999	#_OFSXW[n] n=1...999	R/W	The wear value of length compensation in the direction X. n=1...999: the number of the compensation cell.
#15001...#15999	#_OFSXG[n] n=1...999	R/W	The geometry value of length compensation in the direction X. n=1...999: the number of the compensation cell.
#14001...#14999	#_OFSYW[n] n=1...999	R/W	The wear value of length compensation in the direction Y. n=1...999: the number of the compensation cell.
#19001...#19999	#_OFSYG[n] n=1...999	R/W	The geometry value of length compensation in the direction Y. n=1...999: the number of the compensation cell.
#11001...#11999	#_OFSZW[n] n=1...999	R/W	The wear value of length compensation in the direction Z. n=1...999: the number of the compensation cell.
#16001...#16999	#_OFSZG[n] n=1...999	R/W	The geometry value of length compensation in the direction Z. n=1...999: the number of the compensation cell.
#12001...#12999	#_OFSRW[n] n=1...999	R/W	The wear value of radius compensation. n=1...999: the number of the compensation cell.
#17001...#17999	#_OFSRG[n] n=1...999	R/W	The geometry value of radius compensation. n=1...999: the number of the compensation cell.

Number	Symbol	Attribute	Description
#13001...#13999	#_OFST[n] n=1...999	R/W	The code of tool position. n=1...999: the number of the compensation cell.

Using the macro variables above, value can be assigned to all the compensation cells, and, all the compensation cells can be read out from program.

23.1.19 Workpiece Zero Point Offsets

Values of zero point offset

Number	Symbol	Attribute	Description
#5201...#5220 #100301...#100350	#_WZCMN[n] n=1-50	R/W	Common workpiece zero point offset for each axis
#5221...#5240 #100351...#100400	#_WZG54[n] n=1-50	R/W	Workpiece zero point offset G54 for each axis
#5241...#5260 #100401...#100450	#_WZG55[n] n=1-50	R/W	Workpiece zero point offset G55 for the axis 1...n
#5261...#5280 #100451...#100500	#_WZG56[n] n=1-50	R/W	Workpiece zero point offset G56 for the axis 1...n
#5281...#5300 #100501...#100550	#_WZG57[n] n=1-50	R/W	Workpiece zero point offset G57 for the axis 1...n
#5301...#5320 #100551...#100600	#_WZG58[n] n=1-50	R/W	Workpiece zero point offset G58 for the axis 1...n
#5321...#5340 #100601...#100650	#_WZG59[n] n=1-50	R/W	Workpiece zero point offset G59 for the axis 1...n

Using the macro variables above, value can be assigned to all the zero point offsets for each axis, and, all the zero point offsets for each axis can be read out from program.

Values of misalignment compensation

Number	Symbol	Attribute	Description
#120001	#_WRG54[1]	R/W	Misalignment compensation G54 in the plane XY
#120002	#_WRG54[2]	R/W	Misalignment compensation G54 in the plane ZX
#120003	#_WRG54[3]	R/W	Misalignment compensation G54 in the plane YZ

Number	Symbol	Attribute	Description
#120011	#_WRG55[1]	R/W	Misalignment compensation G55 in the plane XY
#120012	#_WRG55[2]	R/W	Misalignment compensation G55 in the plane ZX
#120013	#_WRG55[3]	R/W	Misalignment compensation G55 in the plane YZ
#120021	#_WRG56[1]	R/W	Misalignment compensation G56 in the plane XY
#120022	#_WRG56[2]	R/W	Misalignment compensation G56 in the plane ZX
#120023	#_WRG56[3]	R/W	Misalignment compensation G56 in the plane YZ
#120031	#_WRG57[1]	R/W	Misalignment compensation G57 in the plane XY
#120032	#_WRG57[2]	R/W	Misalignment compensation G57 in the plane ZX
#120033	#_WRG57[3]	R/W	Misalignment compensation G57 in the plane YZ
#120041	#_WRG58[1]	R/W	Misalignment compensation G58 in the plane XY
#120042	#_WRG58[2]	R/W	Misalignment compensation G58 in the plane ZX
#120043	#_WRG58[3]	R/W	Misalignment compensation G58 in the plane YZ
#120051	#_WRG59[1]	R/W	Misalignment compensation G59 in the plane XY
#120052	#_WRG59[2]	R/W	Misalignment compensation G59 in the plane ZX
#120053	#_WRG59[3]	R/W	Misalignment compensation G59 in the plane YZ

Using the macro variables above, misalignment compensation can be assigned in an arbitrary main plane (but in one main plane only) for each workpiece coordinate system, where the center point of the misalignment compensation is the origin of the coordinate system.

Values of additional zero point offset

Number	Symbol	Attribute	Description
#7001...#7020 #101001...#101050	#_WZP1[n] n=1-50	R/W	Workpiece zero point offset G54.1 P1 for the axis 1...n
#7021...#7040 #101051...#101100	#_WZP2[n] n=1-50	R/W	Workpiece zero point offset G54.1 P2 for the axis 1...n
...	...	...	...
#7941...#7960 #103351...#103400	#_WZP48[n] n=1-50	R/W	Workpiece zero point offset G54.1 P48 for the axis 1...n
#103401...#103450	#_WZP49[n] n=1-50	R/W	Workpiece zero point offset G54.1 P49 for the axis 1...n
#103451...#103500	#_WZP50[n] n=1-50	R/W	Workpiece zero point offset G54.1 P50 for the axis 1...n
...	...	...	...

Using the macro variables above, value can be assigned to all the zero point offsets for each axis, and, all the zero point offsets for each axis can be read out from program.

Values of additional misalignment compensation

Number	Symbol	Attribute	Description
#120061	#_WRP1[1]	R/W	Misalignment compensation G54.1 P1 in the plane XY
#120062	#_WRP1[2]	R/W	Misalignment compensation G54.1 P1 in the plane ZX
#120063	#_WRP1[3]	R/W	Misalignment compensation G54.1 P1 in the plane YZ
#120071	#_WRP2[1]	R/W	Misalignment compensation G54.1 P2 in the plane XY
#120072	#_WRP2[2]	R/W	Misalignment compensation G54.1 P2 in the plane ZX
#120073	#_WRP2[3]	R/W	Misalignment compensation G54.1 P2 in the plane YZ
...	...	...	...

Using the macro variables above, misalignment compensation can be assigned in an arbitrary main plane (but in one main plane only) for each additional workpiece coordinate system, where the center point of the misalignment compensation is the origin of the coordinate system.

Values of dynamic zero point offset

The values of dynamic zero point offset in the table can be written and read using the following variables:

Number	Symbol	Attribute	Description
#5521...#5540	#_FOFS1[n] n=1-50	R/W	Dynamic zero point offset G54.2 P1for each axis
#117051...#117100			
#5541...#5560	#_FOFS2[n] n=1-50	R/W	Dynamic zero point offset G54.2 P2 for each axis
#117101...#117150			
...	...		...
#5661...#5680	#_FOFS8[n] n=1-50	R/W	Dynamic zero point offset G54.2 P8 for each axis
#117401...#117450			

Number of the actual dynamic zero point offset

While the program runs, the number of the valid dynamic zero point offset can be read out using the command below:

Numbe r	Symbol	Attribute	Description
#5500	#_FOFSP	R	The number of the actual dynamic zero point offset (G54.2 Pp P=1-8)
#117000			

Value of the actual dynamic zero point offset

While the program runs, the value of the valid dynamic zero point offset can be read out using the commands below. These values are not identical with the values written in the table of the dynamic zero point offset, but they are values modified (rotated) in accordance with the actual position of the work table(s).

Number	Symbol	Attribute	Description
#5501...#5520	#_FOFSVAL[n]	R	The value of the actual dynamic zero point offset for each axis
#117001...#117050	n=1-50		

23.1.20 Reading Data of Tools Being in Spindle and in Stand-by Magazines

Number	Symbol	Attribute	Description
#8400		R/W	The address of the spindle or the stand-by magazine

The data of tools being in the spindle or in the stand-by magazine can be read using macro variables #8401, #8402, ... by specifying magazine number on variable

#8400 (10, 11, 20, 21, ... writable, readable).

Only the magazine numbers given in brackets can be defined. If there are several spindles or stand-by magazines on the machine, ask the builder of the machine tool for information.
If there is only one spindle and no stand-by magazine on the machine, value will not have to be assigned to the macro variable #8400 and the macro variables #8401, ... will always be related to the tool being in the spindle.

The data that can be read out are the following:

Number	Symbol	Attribute	Description
#8401		R	Data number (serial number of the tool management table)
#8402		R	Tool type number (code T)
#8403		R	Tool life counter value
#8404		R	Maximum tool life value
#8405		R	Warning tool life value
#8406		R	Tool life status
#8407		R	Customer bit-type data
#8408		R	Tool information
#8409		R	H: number of the length compensation cell (for milling machine channel)
#8410		R	D: number of the radius compensation cell (for milling machine channel)
#8411		R	S: spindle speed
#8412		R	F: feed
#8413		R	G: number of the geometry compensation cell (for lathe channel)
#8414		R	W: number of the wear compensation cell (for lathe channel)
#8431		R	Customer data 1
#8432		R	Customer data 2

Number	Symbol	Attribute	Description
#8433		R	Customer data 3
#8434		R	Customer data 4
#8435		R	Customer data 5
#8436		R	Customer data 6
#8437		R	Customer data 7
#8438		R	Customer data 8
#8439		R	Customer data 9
#8440		R	Customer data 10
#8441		R	Customer data 11
#8442		R	Customer data 12
#8443		R	Customer data 13
#8444		R	Customer data 14
#8445		R	Customer data 15
#8446		R	Customer data 16
#8447		R	Customer data 17
#8448		R	Customer data 18
#8449		R	Customer data 19
#8450		R	Customer data 20

The number of customer data can be given in the parameter N2903 No. of Custom Columns. The first customer data is always a bit-type data.
The macro variables above are readable only.
If the spindle or the stand-by magazine selected by the macro variable #8400 is empty (there is no tool in it), the value of the macro variables will be the following:

#8401=0 (data number);

#8402= #8403= ...= #8450= #0 (empty).

Using the macro variables, compensations (H, D) or technological parameters (F, S) assigned to the tool can be called. If, for example, subprogram call is assigned to the code of tool change (M06), the following can be written in the subprogram:

```
...
M6
#8400=10 (1. spindle magazine)
H#8409 D#8410 S#8411 F#8412
...
```

23.1.21 Reading the Data of the Pallet Being in the Working Space and in the Loading-Unloading Point

Number	Symbol	Feature	Description
#8500		R/W	Address of the working space of loading-unloading point

It is the macro variable

#8500 (10, 11 writable, readable)

at which it can be given whether the data of the pallet being in the working space (#8500=10) or the data of the pallet being in the loading-unloading point (#8500=11) have to be read out via macro variables #8501, #8502, ... Only the magazine numbers given in bracket can be defined.
If there is only working space and there is no loading-unloading point, value has not to be given to the macro variable #8500; the macro variables #8501, ... apply always to the pallet being in the working space.
After turn-on, the control starts with the initial value #8500=10.

The data that can be read out are as follows:

Number	Symbol	Feature	Description
#8501		R	Pallet identifier
#8502		R	Status
#8503		R	Priority
#8504		R	Number of executed programs
#8505		R	Number of assigned programs
#8531		R	Custom 1
#8532		R	Custom 2
#8533		R	Custom 3
#8534		R	Custom 4

23.2 Instructions of the Program Language

For describing various instructions, the expression
        #i = <formula>
is used. The <formula> may include arithmetic operations, functions, variables, constants.
In general, the variables #j and #k are referred to in the <formula>.
The <formula> may stand not only in the right side of the assignment statement, but in an NC block, the various addresses may take on a formula too, instead of concrete numerical value or variable.
The order of the operation execution can be influenced by using brackets [,].

23.2.1 Definition or Replacement: #i = #j

The code of this instruction: =
Due to this instruction, the variable #I takes on the value of the variable #j, i.e. the value of the variable #j gets into the variable #i.
The assignment
        #i=#i<operation>#j
is also permitted. An operation will be executed in the value of the variable #I by the variable #j , and the result of the operation will get into the variable #i.
For example:

```
        #100=#100+1
```

means that the new value of the #100 will be its old value plus 1.
Value can be assigned to bit-type variables too:

```
        #1100=1
        #_UO[3]=0
        #_M_SBK=1
```

If the value assigned to the bit-type variable is
        $0 < \text{value} \le 1$,
the value will be taken on as 1. If te value is greater than 1
        $1 < \text{value}$,
the error message ‘2092 Value truncation’ will be sent by the control.

☞ ***Warning! Only writable or writable/readable (W or W/R) variable must stand on the left side of the assignment statement.*** If a value is to be assigned to a variable that is readable (R) only, the error message ‘2161 Macro variable #nnnn is read-only’ will be sent by the control.

23.2.2 Arithmetic Operations

Unary minus: #i = – #j
        The code of the operation: –
        As a result of the operation, the variable #i will be identical with the variable #j in absolute value, but it will have opposite sign.

```
                #100=-#12
```

Addition: #i = #j + #k

The code of the operation: +

As a result of the operation, the variable #i will take on the sum of the variables #j and #k.

```
        #1=#2+3.25
        G0 X[#100+#101]
```

Subtraction: #i = #j – #k

The code of the operation: –

As a result of the operation, the variable #i will take on the difference of the variables #j ans #k.

```
        #100=#100-#102
        G1 Z[25.34-2.48]
```

Multiplication: #i = #j * #k

The code of the operation: *

As a result of the operation, the variable #i will take on the product of the variables #j and #k.

```
        #3=#1*#2
        #100=#101*5.65
```

Division: #i = #j / #k

The code of the operation: /

As a result of the operation, the variable #i will take on the quotient of the variables #j and #k. The value of the variable #k must not be 0, otherwise the error message '2093 Zero divide error' will be sent by the control.

```
        #3=#1/#2
        #100=542.23/#3
```

Producing remainder: #i = #j MOD #k

The code of the operation: **MOD**

As a result of the operation, the variable #i will take on the division remainder of the variables #j and #k. The value of the variable #k must not be 0, otherwise the error message '2093 Zero divide error' will be sent by the control. In the case of

```
        #120=27MOD4
```

the value of the variable #120 will be 3.

Examples of using arithmetic operations

Using the brackets [,], the order of precedence can be influenced.

```
    #100 = #101 + #102
    #100 = #100 - 3
    #100 = [#101 + #102*5.27]/4.1
    G0 X[[#100 + 2]/4] Y100
```

23.2.3 Logic Operations

Logical operations always manipulate the variables as 32-bit signed integer numbers. If logical operation is executed with an originally floating-point variable, e.g. with #100, it will be checked by the control whether the value of the variable #i satisfies the following condition:

$-(2^{32} - 1) \leq \#i < 2^{32}$,

or in decimal format

$-4294967297 \leq \#i < 4294967296$.

If not, the error message ‘2129 Erroneous operation with #’ will be sent by the control.
If the value of the variable is within the these boundaries, the operation will be executed with the 32-bit integer number. If the result of the operation gets to an originally floating-point variable, e.g. to #100, the result will be reconverted into floating-point number by the control.

Negation: #i = NOT #j

The code of the operation: **NOT**

As a result of the operation, at first, the variable #j will be converted into 32-bit fixed-point number. Then, the bitwise negated value of this fixed-point number will be taken at all the 32 bits.

Disjunction: #i = #j OR #k

The code of the operation: **OR**

As a result of the operation, at first the variables #j and #k will be converted into 32-bit fixed-point number.

As a result of the operation, the logic sum of the bitwise values of the variables #j and #k will get to the variable #i, at all the 32 bits. Where there is 0 at the same positional values of the two numbers in both places, at this positional value in the result there will be 0, otherwise 1.

Exclusive or: #i = #j XOR #k

The code of the operation: **XOR**

As a result of the operation, at first the variables #j and #k will be converted into 32-bit fixed-point number.

As a result of the operation, the bitwise values of the variables #j and #k will be summed into the variable #i in such a way so that where there are same numerical values at same positional values, at this positional value in the result there will be 0, but where there are dissimilar numerical values at same positional values, at this positional value in the result there will be 1, at all the 32 bits.

Conjunction: i# = #j AND #k

The code of the operation: **AND**

As a result of the operation, at first the variables #j and #k will be converted into 32-bit fixed-point number.

As a result of the operation, the logic product of the bitwise values of the variables #j and #k will get to the variable #i, at all the 32 bits. Where there is 1 at the same positional values of the two numbers in both places, at this positional value in the result there will be 1, otherwise 0.

Examples of using logic operations

Let us manipulate the 32-bit macro variable #1132 sent to PLC in the following way: the upper 8 bits (31-24) of the variable are to be set in such a way so that the lower bits remain unchanged. To leave the lower 24 bits unchanged and to delete the 8 upper 8 bits, let us have the following bit mask:
The hexadecimal format of the bit mask is:

```
00FFFFFF,
```

and having converted into decimal format, it will be:

```
16777215
```

Let us store the bit mask in the variable #100:

```
#100=16777215
```

Let as store in the variable #100 the value leaving the lower 24 bits of the macro variable

#1132 unchanged but deleted at the upper 8 bits of it:

```
	#101=#100AND#1132
```

Let us write on the variable #102 the bit mask to be set:
The hexadecimal format of the bit mask is:

```
	9B000000,
```

and having converted into decimal format, it will be:

```
	-1694498816
```

Let us store the bit mask in the variable #102:

```
	#102= -1694498816
```

Then, let us write it on the variable #1132 in such a way so that the lower 24 bits remain unchanged:

```
	#1132=#101XOR#102
```

Using brackets accordingly, the following can also be written instead of the four rows above:

```
	#1132=[16777215AND#1132]XOR[-1694498816]
```

23.2.4 Functions

Square root: #i = SQRT #j

The code of the function: **SQRT**

As a result of the operation, the variable #i will take on the square root of the variable #j.

```
		#101=SQRT[#100+4]
```

The value of the variable #j must be only a positive number or 0:

$\#j \geq 0$

If the argument is negative, the error message '2122 Square root from negative value' will be sent by the control.

Sine: #i = SIN #j

The code of the function: **SIN**

As a result of the operation, the variable #i will take on the sine of the variable #j. ***The value of the #j is to be always interpreted in degree***.

```
		G1 X[SIN30-#101]
```

Cosine: #i = COS #j

The code of the function: **COS**

As a result of the operation, the variable #i will take on the cosine of the variable #j. ***The value of the #j is to be always interpreted in degree***.

```
		#102=COS[#1+#2]
```

Tangent: #i = TAN #j

The code of the function: **TAN**

As a result of the operation, the variable #i will take on the tangent of the variable #j. ***The value of the #j is to be always interpreted in degree***.

If the value of the argument #j is

#j=(2n+1)*90°, where n=0, ±1, ±2,...

The error message '2116 Absolute value of the argument is not less then 90' will be sent by the control.

```
		#1=TAN[#2*15.6]
```

Arc sine: #i = ASIN #j

The code of the function: **ASIN**

As a result of the operation, the variable #i will take on the arcsine of the variable #j.

The result will be given in degree, the value of the variable #i will be between +90° and -90°.

```
#101=ASIN[-0.5] (it will be #101=-30)
```

The argument of the variable #j must satisfy the following condition:

$-1 \le \#j \le 1$

Otherwise, the error message '2117 Absolute value of the argument is greater then 1' will be sent by the control.

Arc cosine: #i = ACOS #j

The code of the function: **ACOS**

As a result of the operation, the variable #i will take on the arc cosine of the variable #j. ***The result will be given in degree, the value of the variable #i will be between 0° and 180°***.

```
#101=ACOS[-0.5] (it will be #101=120)
```

The argument of the variable #j must satisfy the following condition:

$-1 \le \#j \le 1$

Otherwise, the error message '2117 Absolute value of the argument is greater then 1' will be sent by the control.

Arc tangent: #i = ATAN #j

The code of the function: **ATAN**

As a result of the operation, the variable #i will take on the arc tangent of the variable #j. ***The result will be given in degree, the value of the variable #i will be between +90° and -90°***.

```
#101=ATAN[-0.5] (it will be #101=-26.565)
```

Exponential with base e: #i = EXP #j

The code of the function: **EXP**

As a result of the operation, the variable #i will take on the #jth power of the exponential growth constant e.

```
#100=EXP1 (it will be #100=2.71828... i.e. "e")
```

Natural logarithm: #i = LN #j

The code of the function: **LN**

As a result of the operation, the variable #i will take on the logarithm to the base of the mathematical constant e of the number #j.

```
#100=LN2.718281828 (it will be #100=0.999... ln(e)=1)
```

If the value of the #j is

$\#j \le 0$,

the error message '2118 Value of the argument is not positive' will be sent by the control.

23.2.5 Conversion Instruction

Generating the absolute value: #i = ABS #j

The code of the function: **ABS**

As a result of the operation, the variable #i will take on the absolute value of the variable #j.

```
#100=ABS[-3.1] (it will be #100=3.1)
#100=ABS[5.25] (it will be #100=5.25)
#100=ABS[0] (it will be #100=0)
```

Rounding down to the absolute value: #i = FIX #j

The code of the function: **FIX**

The fractional part of the variable #j is discarded by the operation, and the variable #i will have the remaining value.
For example:

```
#130=FIX4.8 (it will be #130=4)
#131=FIX[-6.7] (it will be #131=-6)
```

Rounding up to the absolute value: #i = FUP #j
The code of the function: **FUP**
The fractional part of the variable #j is discarded by the operation, and 1 will be added to the absolute value.
For example:

```
#130=FUP12.1 (it will be #130=13)
#131=FUP[-7.3] (it will be #131=-8)
```

23.2.6 Execution Sequence of Complex Arithmetic Operations

The arithmetic operations and the functions listed above can be combined. The ***execution sequence of operations*** or the order of precedence is as follows:
function – multiplicative arithmetic operations – additive arithmetic operations .
For instance, in the example below the execution sequence of the operations is as follows:

```
#110 = #111 + #112 * COS #113
                     ────1───
              ───────2───────
       ──────────3───────────
```

Modifying the execution sequence of operations
Using brackets[and], the execution sequence of operations can be modified.
An example of a bracketing of three-level depth is illustrated below:

```
#120 = COS [ [ [#121 - #122] * #123 + #125] * #126]
                ─────1──────
                ────────2──────────
                ───────────3───────────────
             ─────────────────4───────────────────
       ───────────────────────────5────────────────
```

The numbers indicate the execution sequence of operations. It can be seen that the abovementioned order of precedence apply to execute sequence of operations within the brackets of the same level.

The opening bracket [and the] closing bracket have to be used in pair. If the opening brackets [is less than the closing brackets], the error message '2064 Syntax error' will be sent by the control. If the closing brackets]is less than the opening brackets [, the error message '2121Erroneous terminator #' will be sent by the control.

23.2.7 Conditional Expressions

The conditional expressions known by the program language are the following:

equal to (=):	#i **EQ** #j
not equal to (≠):	#i **NE** #j
greater than (>):	#i **GT** #j
less than (<):	#i **LT** #j
greater than or equal to (≥):	#i **GE** #j
smaller than or equal to (≤):	#i **LE** #j

The variables on the both sides of a conditional expression can be substituted by a formula too. The conditional expressions above can stand after the instructions IF or WHILE.

23.2.8 Unconditional Branch: GOTOn

Due to the instruction **GOTO**n, execution of the program will resume without any condition in the same program, at the block with the sequence number n. The sequence number n can also be replaced by a variable or a formula. ***The sequence number of the block*** to which jumping is to be intended by the use of the instruction GOTO ***must stand at the beginning of the block***.
If the specified block sequence number is not found, the error message '2125 Block not found Nnnnn' will be sent by the control.
Jump can be made either forward:

```
...
GOTO160 (jumping to the block N160)
...
N160 G0 X100
...
```

or backward. In this case an infinite cycle can also be produced:

```
...
N160 G0 X100
...
#1=100
#2=60
...
GOTO[#1+#2] (jumping to the block N160)
...
```

23.2.9 Conditional Branch: IF[<conditional expression>] GOTOn

If [<conditional expression>] put into brackets compulsorily ***is satisfied***, execution of ***the program will resume in the same program, at the block with the sequence number n.***
If [<conditional expression>] is not satisfied, ***execution of the program will resume at the succeeding block.***
If the IF is not followed by condition check, the error message '2064 Syntax error' or '2121 Erroneous terminator=' will be sent by the control, depending on the type of the error.

For example:

```
      ...
      #100=52.28
      #101=16.87
      ...
      IF[#100GT#101] GOTO210 (jumping to the block N210)
      ...
      N210 G0 X0 Y100 Z20
      ...
```

23.2.10 Conditional Instruction: IF[<conditional expression>] (THEN)<instruction>

If [<conditional expression>] is satisfied, the instruction following the THEN will be executed.
If [<conditional expression>] is not satisfied**, **execution of the program will resume at the succeeding block.
The word THEN can be omitted in the instruction, execution of the instruction line

IF[<conditional expression>] instruction

will be the same.

```
      ...
      #100=0
      #101=0
      #1=20
      ...
      IF[#1EQ20] THEN#100=3.15 (it will be #100=3.15)
      IF[#100GT3.15] #101=5 (it will remain #101=0))
      ...
```

23.2.11 Iteration: WHILE[<conditional expression>] DOm ... ENDm

As long as [<conditional expression>] is satisfied, the blocks from DOm to ENDm will be executed repeatedly. I.e. the control checks whether the condition is satisfied: if yes, the program part between DOm and ENDm will be executed and then, due to the instruction ENDm, the program returns to checking once again the condition following the command WHILE.
If [<conditional expression>] is not satisfied**, **execution of the program will resume at the block following ENDm.
If WHILE [<conditional expression>] is omitted, i.e. the cycle is described by the instructions DOm ... ENDm***, the program part between Dom and ENDm will be executed for infinite period of time.***

The possible values of m: 1, 2, 3.

If m>3, the error message '2002 DO data out of range' will be sent by the control.
If not a condition check but an assignment follows the command WHILE, the error message '2121 Erroneous terminator=' will be sent by the control.
If, in the same block, the instruction WHILE is not followed by DO, the error message '2110 WHILE without DO' will be sent by the control.
If there is no Dom before the instruction ENDm, the error message '2125 Block not found: Dom' will be sent by the control.

Let us produce a hollow using the following data: depth in the direction Z is 5.4 mm; width in the direction X is 21 mm; the zero point located at the center of the hollow in the direction X and at the top of the hollow in the direction Z. Let depth of cut be 0.2 mm. The task can be solved using cycle organization:

```
#1=0           (Target position in the direction Z)
#2=10          (Target position in the direction X)
G0 X#2         (Motion to the starting point in the
               direction X)
Z[#1+1]        (Motion to the starting point in the
               direction Z)
WHILE[#1GT-5.39] DO1     (Cycle until depth of 5.4 mm is
                         reached in the direction Z)
#1=#1-0.2      (Depth of cut of 0.2 mm in the direction Z)
#2=-#2         (Reversing the motion direction X)
G1 Z#1 F100    (Infeed in the direction Z)
X#2 F500       (Milling in the direction X)
END1
G0 Z20
...
```

<u>The rules of cycle organization are as follows:</u>

– The instruction Dom has to be specified before the instruction ENDm:

```
:
END1
:
:                    WRONG
:
DO1
```

In the case above, the error message ‘2125 Block not found: Dom’ will be sent by the control at the block END1.

– The instructions DOm and ENDm has to stand in pair:

```
:
DO1
:
DO1                  WRONG
:
END1
:
```

or

```
:
DO1
:
END1                 WRONG
:
END1
:
```

– It is allowed to use the same identification number several times:

```
:
DO1
:
END1
:
:                    RIGHT
:
DO1
:
END1
:
```

– The pairs DOm ... ENDm can be nested into each other up to three levels at most:

```
:
DO1
:
DO2
:
DO3
:
:                    RIGHT
:
END3
:
END2
:
END1
:
```

– The pairs DOm ... ENDm must not overlap each other:

```
:
DO1
:
DO2
:
:                    WRONG
:
END1
:
END2
```

– It is allowed to branch from inside the cycle to outside the cycle:

```
    :
    DO1
    :
    GOTO150
    :
    :                    RIGHT
    :
    END1
    :
    N150
    :
```

– It is allowed to enter the cycle from outside:

```
    :
    GOTO150
    :
    DO1
    :
    :
    :
    N150
    :
    END1
    :

or

    :
    DO1
    :
    N150
    :
    :
    :
    END1
    :
    GOTO150
    :
```

– Calling subprograms or macros from inside the cycle is possible. Inside the subprogram or the custom macros, the cycles can again be nested into each other up to three levels:

```
:
DO1
:
M98...              RIGHT
:
G65...              RIGHT
:
G66...              RIGHT
:
G67...              RIGHT
:
END1
:
```

23.2.12 Indirect Axis Address Specification

The instruction

#i=AXNUM[<axis address>]

makes it possible to query the number of an axis in a given channel by the address of that axis. If it is intended to query the axis number for a non-existent axis address, the error message '2017 Illegal address <axis name>' will be sent by the control.
Using the instruction

AX[<axis number>]=

a command can be issued indirectly, i.e. not by the address of the axis, but by the number of the axis. If there is no the axis of the specified number in the channel, the error message '2018 Bad physical axis number: n' will be sent by the control.
For example, instead of the block

```
G0 X10 Y20 Z30
```

the following instruction line can be written:

```
#101=AXNUM[X]       (Querying the number of the axis X)
#102=AXNUM[Y]       (Querying the number of the axis Y)
#103=AXNUM[Z]       (Querying the number of the axis Z)
G0 AX[#101]=10 AX[#102]=20 AX[#101]=30
```

☞ ***Warning!*** If multi-character axis addresses are used, the axis address will not have to be AX or AXN.

23.2.13 Data Output Commands

The following data output commands are known by the control:

POPEN	opening a peripheral
FOPEN	opening a file
BPRNT	binary data output
DPRNT	decimal data output
PCLOS	closing a peripheral
FCLOS	closing a file

These data output commands can be used to output characters and variable values. Outputting occurs in the memory of the control. They can be used, for example, for storing measurement results, logging etc.

Opening a peripheral: POPENn

Prior to issuing a data output command, it is necessary to open the appropriate peripheral through which data output will be executed. The number Number is used for selecting the appropriate peripheral:

n = 31 the memory of the control

On opening the peripheral, one character % is output to the peripheral so each data output begins with one character %.

After the command POPEN, a filename has to be output by the command DPRNT. If the file already exists, the old one will be overwritten without asking; if it does not exist yet, a new one will be open. The file gets to the folder where the program runs and takes on the extension of the program.

For example:

```
...
#100=4567
POPEN31
DPRNT[O#100[4]]     (The filename will be O4567)
...
```

or

```
...
POPEN31
DPRNT[ABC]          (The filename will be ABC)
...
```

Closing a peripheral: PCLOSn

The peripheral opened by the command POPEN has to be closed by the command PCLOS. After the command PCLOS it is necessary to give the number of the peripheral to be closed. In our case:

```
...
PCLOS31
...
```

On closing, one character % is also output to the peripheral, i.e. each data output is closed with one character %.

Opening a file: FOPENn Ppppp, FOPENn <filename>

Prior to issuing a data output command, it is necessary to open a file where the data are to be written.

Opening a file by program number

The command

FOPENn P(program number)

opens the file the number of which is given at the address P (program number).

The rules of programming the program number given at the address P are dealt with in the subsection **14.4.1 Identification of Programs in Memory. The Program Number (O)** on page 117. ***The rules of extension and in-memory location of the files opened by program number*** are the same as those of subprograms. See the subsection **14.4.2 Calling a Subprogram (M98)** on page 117.

For example:

```
        ...
        FOPEN1 P12          (The filename will be O0012)
        ...
```

Opening a file by filename

The command

FOPENn <filename>

opens the file the name of which is given between the symbols < and >.

The rules of filename specification are dealt with in the subsection **14.4.1 Identification of Programs in Memory. The Program Number (O)** on page 117. ***The rules of extension and in-memory location of the files opened by filename*** are the same as those of subprograms. See the subsection **14.4.2 Calling a Subprogram (M98)** on page 117.

For example:

```
        ...
        FOPEN1 <data.txt>   (The filename will be data.txt)
        ...
```

Types of opening a file

The type of opening a file can be given by the number written at the place of 'n':

n = 1: Creating a new file

If the file already exists, the error message '2144 The file (Ooooo) already exists' or '2146 The given file already exists' will be sent by the system, and the program run does not continue.

n = 2: Creating a new file in any case

If the file does not exist, it will be created; if the file exists, it will be open and its content will be deleted. The contents output by the command BPRNT or DPRNT will be written from the beginning of the file.

n = 3: Opening a file for attachment

If the file does not exist, the error message '2147 The file (Ooooo) not found' or '2132 The file not found' will be sent by the system, and the program run does not continue. If the file already exists, it will be open and, ***beginning from the end of the file***, the contents output by the command BPRNT or DPRNT ***will be attached***.

n = 4: Creating a file or opening it for attachment

If the file ***does not exist*** yet, it ***will be created***; ***if*** the file already ***exists***, it ***will be open*** and, ***beginning from the end of the file***, the contents output by the command BPRNT or DPRNT ***will be attached***.

n = 5: Opening a file together with content deletion

If the file does not exist, the error message '2147 The file (Ooooo) not found' or '2132 The file not found' will be sent by the system, and the program run does not continue. If the file already exists, ***it will be open and its content will be deleted***. The contents output by the command BPRNT or DPRNT will be written from the beginning of the file.

Closing a file: FCLOS

The opened file can be closed by the command FCLOS. For writing, one file can be created or opened simultaneously.

Binary data output: BPRNT[...]

The format of the command BPRNT is as follows:

```
BPRNT[ a #b [c] ... ]
       |  |  |
       |  |  └──────── number of digits after the decimal point
       |  └─────────── variable
       └────────────── character
```

The command send characters in ASCII code, variables in binary format.

– Characters that can be sent are the following:
alphabetical characters: A, B, ..., Z
numerical characters: 1, 2, ..., 0
special characters: *, /, +, –
Instead of the character * (asterisk), the code of the space will be sent by the control.

– Values of the variables are output by the control at 4 bytes, i.e. at 32 bits, beginning from the most significant byte. After the number of variables, the number of digits after the decimal point has to be given in brackets [ ]. In this case, the control converts the floating-point value of the variable into such fixed-point value in which the number of significant decimal digits will be the value given in brackets [ ]. The possible values of c are: 1, 2, ..., 8. For example, if
#120 = 258.647673 és [3] ——— 258648=0003F258h will be output.

– A vacant variable will be output with binary code 00000000h

– At the end of data output, one character of carriage return (CR) and one character of line feed (LF) will automatically be output by the control.

Example

```
#110=318.49362
#120=0.723415
#112=23.9
BPRNT[C*/X#110[3]Y#120[3]M#112[0]]
```

After rounding and hexadecimal conversion, the values of the variables #110, #120 and #112 will be the following:

```
#110=318.49362    ————————    318494=0004DC1Eh
#120=0.723415     ————————    723=000002D3h
#112=23.9         ————————    24=00000018h
```

The characters to be output are the following:

```
    7 6 5 4 3 2 1 0

    0 1 0 0 0 0 1 1    C
    0 0 1 0 0 0 0 0    Space
    0 0 1 0 1 1 1 1    /
    0 1 0 1 1 0 0 0    X
    0 0 0 0 0 0 0 0    00
    0 0 0 0 0 1 0 0    04
    1 1 0 1 1 1 0 0    DC
    0 0 0 1 1 1 1 0    1E
    0 1 0 1 1 0 0 1    Y
    0 0 0 0 0 0 0 0    00
    0 0 0 0 0 0 0 0    00
    0 0 0 0 0 0 1 0    02
    1 1 0 1 0 0 1 1    D3
    0 1 0 0 1 1 0 1    M
    0 0 0 0 0 0 0 0    00
    0 0 0 0 0 0 0 0    00
    0 0 0 0 0 0 0 0    00
    0 0 0 1 1 0 0 0    18
    0 0 0 0 1 1 0 1    Carriage Return
    0 0 0 0 1 0 1 0    Line Feed
```

Decimal data output: DPRNT[...]
The format of the command DPRNT is as follows:

```
DPRNT[ a #b [c d] ... ]
       |  |  |  |
       |  |  |  └─ number of digits after the decimal point
       |  |  └──── number of digits before the decimal point
       |  └─────── variable
       └────────── character
```

All the characters and numbers will be output in ASCII code.
– See the command **BPRNT** for the rules applied to outputting the characters.
– For outputting the values of the variables, the number of decimal integers and fractions in which the variable is to be output has to be given. The numeral digits have to be given in brackets [ ].
For specification of numerical digits, the condition 0 < c + d < 16 has to be specified. Outputting the numbers begins from their highest place value.
– When number is output, the negative sign (−) will also be output.
– If the decimal point is defined (d > 0), all the zeros, including the following zeros too, will be output together with the decimal point (.).
– If d=0, or d is not given, neither decimal point, nor zero will be output by the control.
– If the bit #0 PNT of the parameter N1757 Print Contr is
=0, the code of space will be output in the positions of the sign + and the zeros;
=1, the sign + and the leading will not be output.
– A vacant variable will be output with code 0.

– At the end of data output, one character of carriage return (CR) and one character of line feed (LF) will automatically be output by the control.

Example

```
#130=35.897421
#500=-150.8
#10=14.8
DPRNT[X#130[53]Y#500[53]T#10[20]
```

After rounding, the values of the variables #130, #500 and #10 will be the following:

#130=35.897421	————	35.897
#500=–150.8	————	150.8
#10=14.8	————	15

Data output at the state #0 PNT=0 of the parameterN1757 Print Contr:

```
7 6 5 4 3 2 1 0

0 1 0 1 1 0 0 0     X
0 0 1 0 0 0 0 0     Space
0 0 1 0 0 0 0 0     Space
0 0 1 0 0 0 0 0     Space
0 0 1 0 0 0 0 0     Space
0 0 1 1 0 0 1 1     3
0 0 1 1 0 1 0 1     5
0 0 1 0 1 1 1 0     Decimal point (.)
0 0 1 1 1 0 0 0     8
0 0 1 1 1 0 0 1     9
0 0 1 1 0 1 1 1     7
0 1 0 1 1 0 0 1     Y
0 0 1 0 1 1 0 1     Negative sign (-)
0 0 1 0 0 0 0 0     Space
0 0 1 0 0 0 0 0     Space
0 0 1 1 0 0 0 1     1
0 0 1 1 0 1 0 1     5
0 0 1 1 0 0 0 0     0
0 0 1 0 1 1 1 0     Decimal point (.)
0 0 1 1 1 0 0 0     8
0 0 1 1 0 0 0 0     0
0 0 1 1 0 0 0 0     0
0 1 0 1 0 1 0 0     T
0 0 1 0 0 0 0 0     Space
0 0 1 1 0 0 0 1     1
0 0 1 1 0 1 0 1     5
0 0 0 0 1 1 0 1     Carriage Return
0 0 0 0 1 0 1 0     Line Feed
```

Data output at the state #0 PNT=1 of the parameterN1757 Print Contr:

```
7 6 5 4 3 2 1 0

0 1 0 1 1 0 0 0     X
0 0 1 1 0 0 1 1     3
0 0 1 1 0 1 0 1     5
0 0 1 0 1 1 1 0     Decimal point (.)
0 0 1 1 1 0 0 0     8
0 0 1 1 1 0 0 1     9
0 0 1 1 0 1 1 1     7
0 1 0 1 1 0 0 1     Y
0 0 1 0 1 1 0 1     Negative sign (-)
0 0 1 1 0 0 0 1     1
0 0 1 1 0 1 0 1     5
0 0 1 1 0 0 0 0     0
0 0 1 0 1 1 1 0     Decimal point (.)
0 0 1 1 1 0 0 0     8
0 0 1 1 0 0 0 0     0
0 0 1 1 0 0 0 0     0
0 1 0 1 0 1 0 0     T
0 0 1 1 0 0 0 1     1
0 0 1 1 0 1 0 1     5
0 0 0 0 1 1 0 1     Carriage Return
0 0 0 0 1 0 1 0     Line Feed
```

☞ ***Notes***:

– The sequence of data output commands is fixed: at first, the appropriate file has to be open by the command POPEN or FOPEN, then data output can be executed by the command BPRNT or DPRNT, and finally, the opened file has to be closed by the command PCLOS or FCLOS.

– Opening and closing the file can be given in any point of the program. For example, the file can be open at the beginning of the program and can be closed at the program end, while data can be output in any part of the program between these two commands.

– The command M30 or M2 executed during data output will interrupt the data transfer. To avoid this, prior to execution of the command M30 it is necessary to wait during data transfer.

23.3 Calling the Macros, System Macros and System Subprograms

Macro call is similar to subprogram call with the difference that ***macros***, contrary to subprograms, ***can have input variables***, i.e. ***arguments***.
Both macros and subprograms can be called either by their program number or by their filename.

Those macros and subprograms are called ***system macros*** and ***system subprograms*** the call of which is related to an ***address assigned in the parameter*** (for example G, M etc.). ***Specific rules apply to filenames of system macros and system subprograms and to their location in the memory***.

A macro call can be multi-level too. A macro can be called from a subprogram, and a subprogram can be called from a macro. The ***maximum*** aggregated level of calling subprograms and macros is ***16***.

Return from a macro occurs ***by the code***
M99.

One-shot and modal macro calls are distinguished. The modal macro calls can be deleted by the code
G67.

Argument assignment
Arguments can be assigned to a macro program. ***Arguments are specific numerical values assigned to definite addresses that, during macro call, will be stored in the appropriate local variables (#1, #2, ..., #33)***. A macro program (special subprogram) can use these local variables, i.e. a macro call is such a special subprogram call, in which the main program can transfer variables (parameters) to the subprogram.
There are two possible argument assignments:

Argument assignment No.1
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
In the case of argument assignment No.1, the parameters are transferred to the macro by using the letters of the English alphabet. Some of the letters above (***G, P, L***) can be monopolized by certain calls for specific purpose; in that case, these letters cannot be used for parameter transfer. Filling the addresses can be made in arbitrary order; it is not necessary to write them into the calling block in alphabetical order.

Argument assignment No.2
A B C I1 J1 K1 I2 J2 K2 ... I10 J10 K10
In the case of argument assignment No.2, the arguments are transferred to the macro by using the addresses I, J and K, in addition to the addresses A, B and C. At the addresses A, B and C, maximum 10 different argument group can be assigned. If, in a block, several arguments are assigned at the same address, the variables will take on the appropriate value in the order of assignment.

Relations between the local variables and the arguments are as follows:

l v	*1. a a*	*2. a a*
#1	A	A
#2	B	B
#3	C	C
#4	I	I1
#5	J	J1
#6	K	K1
#7	D	I2
#8	E	J2
#9	F	K2
#10	G	I3
#11	H	J3
#12	L	K3
#13	M	I4
#14	N	J4
#15	O	K4
#16	P	I5
#17	Q	J5
#18	R	K5
#19	S	I6
#20	T	J6
#21	U	K6
#22	V	I7
#23	W	J7
#24	X	K7
#25	Y	I8
#26	Z	J8
#27	–	K8
#28	–	I9
#29	–	J9
#30	–	K9
#31	–	I10
#32	–	J10
#33	–	K10

where:	*lv*:	local variable
	1.a a:	argument assignment No.1,
	2.a a:	argument assignment No.2.

The indexes following the addresses I, J and K show the sequence of argument assignment. Decimal point and sign can also be transferred at the addresses.

Handling the address N

The ***first address N*** written in the block is interpreted as ***block number***, the ***second address N***, however, ***will be recorded*** in the local variable #14 as ***argument***:

```
     /N130 X12.3 Y32.6 N250
          N130: block number
          #24=12.3
          #25=32.6
          #14=250 (argument)
```

If the address N had already been recorded as argument, the succeeding reference to the address N will result in the error message '2017Illegal address N'.

Mixed argument assignment

The argument assignment No.1 and the argument assignment No.2 can exist together within a block, the control accepts it. Error message will be sent in the case when a variable of a given number is referred twice. For example:

```
     A2.12 B3.213 J36.9 J-12 E129.73 P2200
```

The local variables take on the address values in the following sequence:

```
          A: #1=2.12
          B: #2=3.213
          J: #5=36.9 (first J)
          J: #8=-12 (second J)
          E: #8=ERROR, the variable #8 had already been filled
```

In this example, the second address J (with the value –12) assigned a value to the variable #8. Since the value of the address E will also be taken on by the variable #8, the error message

‘2017 Illegal address E’ will be sent by the control.
But, if the sequence of assigning the second address J and the address E is interchanged,

```
      A2.12 B3.213 J36.9 E129.73 J-12 P2200
```

error message will not be sent, and the value of the address J will be written in the variable #11 of the succeeding J:

```
            A: #1=2.12
            B: #2=3.213
            J: #5=36.9 (first J)
            E: #8=129.73
            J: #11=-12 (it gets to the variable of the third J
                 because the variable #8 is already occupied)
```

☞ ***Warning! Only one-character addresses can be used for argument assignment***, it means that ,C ,R ,A and multi-character axis addresses and spindle addresses not!

Argument levels
A macro call, like a subprogram call, ***can also be multi-level***, but the maximum aggregated level of calling subprograms and macros can be 16.
Therefore, the local variables #1 ... #33 are multi-level too. The level of the local variables belonging to the main program is 0 (zero level), then macro calls fill the levels 1, 2 etc. in sequence, up to the maximum level 16.
A subprogram or calling the system subprogram does not change the level of local variables.
After return from macro call, the local variables of the given level will be eliminated, they will be deleted to empty by the #0. ***The local variables of the main program*** will be eliminated ***at the end of the program***.

Macro call by program number
By programming
P(program number)
the macro program with the number defined at the address P (program number) will be called.
The rules of programming the program number given at the address P are dealt with in the subsection **14.4.1 Identification of Programs in Memory. The Program Number (O)** on page 117. ***The rules of extension and in-memory location of the macros defined by program number*** are the same as those of subprograms. See the subsection **14.4.2 Calling a Subprogram (M98)** on page 117.
Macro call by filename
☞ ***Warning! Only G65-type*** non-modal ***macros can be called*** by filename, G66-type and G66.1-type modal macros not!
By programming
<filename>
the macro program with the filename given between the symbols < and > will be called.
The rules of filename specification are dealt with in the subsection **14.4.1 Identification of Programs in Memory. The Program Number (O)** on page 117. ***The rules of extension and in-memory location of the macros defined by filename*** are the same as those of subprograms. See the subsection **14.4.2 Calling a Subprogram (M98)** on page 117.

Calling the system macros and subprograms and their in-memory location

System macros and subprograms ***can only be called at the address specified in parameter***. The rules applied to the storage of system macros and subprograms in memory are stricter than those of macros and subprograms.
In the memory, the system macros and subprograms have to be stored ***in separate folder for each channel***, namely, ***in the folder SystemMacros of the directory Programs***:

..\Programs\SystemMacros\Channel1\
..\Programs\SystemMacros\Channel2\

..\Programs\SystemMacros\Channel8\

Their filename has to begin with the letter O which has to be followed by ***four decimal digits. The extension of the filename has to be .nct.*** For example***:***

O9010.nct

23.3.1 Simple Macro Call (G65)

Due to the instructions

G65 P(program number) L(repetition number) <argument assignment>
G65 <filename> L(repetition number) <argument assignment>

the macro program with the number defined at the address P (program number) or with the filename given between the symbols < and > will be called sequentially and repeatedly; the repetition number is given at the address L.
It is not a modal call.

Rules of argument assignment in the case of call G65

Both forms of argument assignment can be used.
In the case of the call G65, the addresses G, P and L cannot be used for argument transfer, but the values specified will be written in the appropriate local variable.

Multiple call G65

If macro is called from macro again, the level of local variables will grow in accordance with the level of macro.

```
Main program         Macro      Macro      Macro      Macro
Level 0              Level 1    Level 2    Level 3    Level 4
                     O______    O______    O______    O______


G65 P                G65 P      G65 P      G65 P


                     M99        M99        M99        M99

Local variables
Level 0              Level 1    Level 2    Level 3    Level 4
#1                   #1         #1         #1         #1
                     :          :          :          :
#33                  #33        #33        #33        #33
```

When the first macro is called, the local variables of the main program from #1 to #33 will be stored, and, at the level 1, the local variables will take on the argument values specified on call. In the case of a further macro call from the level 1, the local variables of the level 1 from

#1 to #33 will be stored, and, at the level 2, the local variables will take on the argument values specified on call. In the case of multiple call, the local variables of the previous level will be stored, and, at the succeeding level, the local variables will take on the argument values specified on call. In the case of M99, when return from the called macro to the calling program occurs, the stored local variables of the previous level will be reset to the state in which they were stored on call.

A hole pattern is to be produced using call G65. The code of drilling is to be specified at the address ‘A’, the other addresses have to be filled in the way used in the case of drilling cycles.

Let the ***main program*** be the following:

```
G54 G17 X0 Y0 Z20
...
G65 P300 A81 Z-2 R2 F300 S500 M3 (center punching by G81)
G65 P300 A83 Z-30 R2 Q6 E1 F100 S1000 (drilling by G83)
G65 P300 A84.2 Z-30 R2 S1000 F1000 (thread tapping by
G84.2)
G0 Z20
...
```

In the macro O0300, the arguments of the call will be received by the local variables, i.e. data related to the spindle:

speed S#19
direction of rotation M#13;

input parameters of drilling:

drilling cycle code A#1
hole depth Z#26
coordinate of the point R R#18
feedrate F#9
infeed Q#17
distance to the point of approaching E#8.

Then, listing the coordinates X, Y of the hole follows.

The body of the ***macro O0300*** executing the drilling is as follows:

```
S#19 M#13                          (Starting the spindle)
G#1 X0 Y0 Z#26 R#18 F#9 Q#17 E#8 (Setting the drilling
cycle)
X100
Y100
X0
X50 Y50
G80
M99
```

23.3.2 Macro Modal Call after Each Motion Command (G66)

Due to the instruction

G66 P(program number) L(repetition number) <argument assignment>

the ***macro program*** with the number defined at the address P (program number) ***will be called*** sequentially and repeatedly ***after execution of each motion command***; the repetition number is given at the address L.

It is a modal call.

The assigned macro will be called until the command

G67

is issued to cancel the modal call.

Rules of argument assignment in the case of call G66

Both forms of argument assignment can be used.

In the case of the call G66, the addresses G, P and L cannot be used for argument transfer, but the values specified will be written in the appropriate local variable.

Example

In a given segment of the part program, a hole has to be produced after each motion:

Main program

```
...
G66 P1250 Z-100 R-1 X2 F130 (Z: the bottom point of the
     hole, R: the point R of the hole, X: dwell time, F:
     feedrate)
N100 G91 G0 X100
N110 Y30
...
N180 X150
G67
```

From the block N110 up to and including the block N180, the macro program O1250 will be called at the end of positioning, and the drilling operation written there will be executed with the input parameters specified in the block G66.

Macro program

```
O1250
G0 Z#18        (Rapid traverse positioning in the
               direction Z to the position -1 given at the
               address R)
G1 Z#26 F#9    (Drilling to the bottom point -100 given at
               the address Z, at the feedrate 130 mm/min
               given at the address F)
G4 P#24        (Dwell at the bottom of the hole for 2 s
               given at the address X)
G0 Z-[#18+#26](Retracting the tool to the initial point)
M99            (Return to the calling program)
```

The state G91 will be inherited from the block N100 of the main program by the macro program O1250, for this reason, each positioning will be executed incrementally.

Multiple call G66

In the case of multiple call of G66-type macros, after execution of each motion block, ***at first***

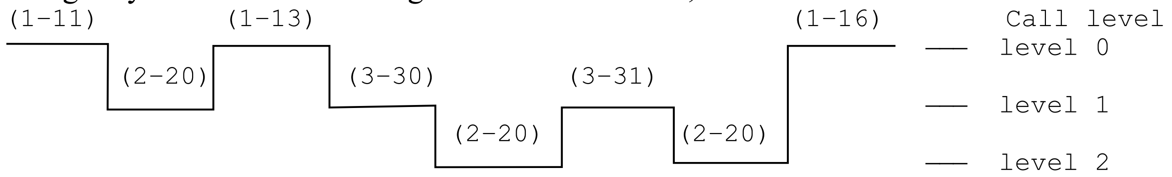
the macro that was called last will be called, and the macros that were called previously will be called from this macro, in reverse order. Let us see the example below:

```
O0001
...
N10 G66 P2
N11 G1 G91 Z10          (1-11)
N12 G66 P3
N13 Z20            (1-13)
N14 G67            (G66 P3 call deletion)
N15 G67            (G66 P2 call deletion)
N16 Z-5            (1-16)
...

O0002
N20 X4             (2-20)
N21 M99

O0003
N30 Z2             (3-30)
N31 Z3             (3-31)
N32 M99
```

Taking only the blocks containing motion into account, the execution order will be as follows:



The first of the numbers in parentheses means the number of the program being under execution, and the second one means the number of the block being under execution.

Each call G66 must be deleted by separate instruction G67. The first instruction G67 deletes the call G66 that was called last, and then, the other calls will be deleted by a succeeding instruction G67, in reverse order. The instruction G67 given in the block N14 deletes the macro called in the block N12 (O0003), and the instruction G67 given in the block N15 deletes the macro called in the block N10 (O0002).

23.3.3 Macro Modal Call from Each Block (G66.1)

Due to the instruction

G66.1 P(program number) L(repetition number) <argument assignment>

all the blocks following it will be interpreted as an argument assignment, the ***macro program*** with the number defined at the address P (program number) ***will be called*** sequentially and repeatedly ***for each block***; the repetition number is given at the address L.
The effect produced by the instruction is the same as if each block was a macro call G65:

```
G66.1 P L
X Y Z = G65 P L X Y Z
M S = G65 P L M S
X = G65 P L X
G67
```

It is a modal call.
The assigned macro will be called until the command

G67

is issued to cancel the modal call.

Rules of argument assignment

1. ***In the block executing start*** (where G66.1 P L is programmed):
   In the call G66.1, the addresses G, P and L cannot be used for argument transfer, but the values specified will be written in the appropriate local variable.
2. ***In the blocks following the instruction G66.1***:
   The addresses G, P and L can also be used.
   The address G (G: #10) ***can be used with the restriction*** that reference to ***only one address G in a block*** is accepted by the control; if several addresses G are programmed, the error message '2017 Illegal address G' will be sent by the control.

Rules of argument assignment in the case of G66.1

The assigned macro will already be called from that block where the code G66.1 was given, taking the rules of argument assignment in the item 1 into account.
From the block following the code G66.1 to the block containing the code G67, each NC block will resulted in macro call in accordance with the rules of argument assignment in the item 2. Macro will not be called in the case of vacant block, e.g. N1240 where there is only a reference to an address N, or it will not be called from the block containing macro instruction.

Multiple call G66.1

In the case of multiple call of G66.1-type macros, ***at first the macro called last will be called on read-in of each block***, handling the addresses of this block as argument; and then, reading in and handling the blocks of this macro, the macro specified one-ahead will be called.
Each call G66.1 must be deleted by separate instruction G67. The first instruction G67 deletes the call G66.1 that was called last, and then, the other calls will be deleted by a succeeding instruction G67, in reverse order.

The following program part had been written for the plane XY, at vertical position of the machine head.

```
...
G1 X40 Y30 F1000
G3 X0 Y50 I-40 J-30
G1 Y0
...
```

In the meantime, the machine head had been mounted into horizontal position. In order that the program written for the plane XY has not to be written again, the axes Y and Z have to be inverted, and direction of the axis X has to be reversed so that the coordinate system remains right-handed. By calling the macro O0400, the direction of motion along X will be reversed and the axes Y and Z will be inverted:

Main program:

```
...
G66.1 P400            (Call for inverting macro)
G18 X0 Y0 Z-5         (Plane designation, positioning)
```

```
    G1 X40 Y30 F1000         (O0400 call)
    G3 X0 Y50 I-40 J-30      (O0400 call)
    G1 Y0                    (O0400 call)
    G67                      (Deletion of modal call)
    ...
```

Macro:

```
    IF[#10EQ66.1] GOTO10 (avoiding recursive call)
    G#10 X-#24 Y#26 Z#25 I-#4 K#5 R#18 F#9 (changes)
    N10 M99
```

The macro O0400 will already be called by the block G66.1.P400. In this case, the code 66.1 will be received at the variable #10. By the first call, return will immediately be executed. The second block of the macro will receive and process the arguments: It will invert the axes Y and Z, and reverse the direction X.

23.3.4 System Macro Call by Codes G Given in Parameter

For macro call, single codes G or arrays of codes G given in parameter can be assigned for each channel.

Assigning 10 single codes G for system macro call

In the parameter field, ***for each channel, maximum 10 different codes G can be assigned,*** for which macro call can be initiated. In this case, instead of the instruction line

Nn G65 Pp <argument assignment>

the instruction line

Nn **Gg** <argument assignment>

has to be written. In the parameter field it is necessary to set which calling G code has to call which program number. The codes G65, G66, G66.1 and G67cannot be assigned for this purpose.

These parameters are the following:

N1704 G(9010): the code G calling the program with the name O9010.nct
N1705 G(9011): the code G calling the program with the name O9011.nct
:
N1713 G(9019): the code G calling the program with the name O9019.nct

If 0 is written in the parameter, the macro with the given program number will not be called.
If ***macro call*** is to be initiated ***by G0***, ***1000*** has to be written in the parameter.

Assigning an array of codes G for system macro call

By using the parameters below, ***an array of codes G can be assigned*** for macro call, for each channel.

In the parameter N1714 Start G Macro, the initial number of the array of codes G is specified by decimal integer number.

In the parameter N1715 Start Prg No, the program number of the macro belonging to the code G given in the parameter Start G Macro is specified.

In the parameter N1716 No. of G Codes, the number of elements of the array of codes G is specified.

If No. of G Codes=0, for these codes G macro will not be called.

Example

Let, for example, the initial code of the array be G2000. Therefore, Start G Macro=2000 has to be set.
If the code G2000 calls the program O3400, then Start Prg No=3400.
In the parameter No. of G Codes, the number of codes G belonging to the array can be specified. If there is 50 codes in the array, then No. of G Codes=50.
So, relation between the codes G and the program numbers is as follows:

Code G 1	G2000	➜	Program number 1	O3400
Code G 2	G2001	➜	Program number 2	O3401
Code G 3	G2002	➜	Program number 3	O3402
....				
Code G 50	G2049	➜	Program number 50	O3449

Assigning an array of codes G with decimal point for system macro call

By using the parameters below, such an array of codes G can be assigned for macro call, for each channel, the code of which contains decimal point and one decimal place.

In the parameter N1717 Start Dec G Macro, the initial number of the array of codes G is specified with decimal point and one decimal place.

In the parameter N1718 Start Prg No. Dec G, the program number of the macro belonging to the code G given in the parameter Start Dec G Macro is specified.

In the parameter N1719 No. of Dec G Codes, the number of elements of the array of codes G containing decimal point is specified.

If No. of Dec G Codes=0, for these codes G macro will not be called.

Example

If the starting code of the group is, for example, G310.5, the parameter has to be filled as follows: Start Dec G Macro=310.5.
If the code G310.5 calls the program O4000, then Start Prg No. Dec G=4000.
In the parameter No. of Dec G Codes, the number of G codes belonging to the group can be given. If there are 10 codes, then No. of Dec G Codes=10.
So, relation between the codes G and the program numbers is as follows:

Code G 1	G310.5	➜	Program number 1	O4000
Code G 2	G310.6	➜	Program number 2	O4001
Code G 3	G310.7	➜	Program number 3	O4002
....				
Code G 10	G311.4	➜	Program number 10	O4009

Making calls for code G modal

If ***negative value is assigned*** to single codes G in the case of single calls, and to the ***parameters*** N1714 Start G Macro or N1717 Start Dec G Macro in the case of array calls, ***modal call will be generated by the assigned code G or group of codes G.*** For example, if G(9011)=–120, the instruction G120 will result modal call in the program. The type of the call will be determined by the following parameter state :

N1755 Macro Contr #0 MEQ=0: the type of the call G is G66;

N1755 Macro Contr #0 MEQ=1: the type of the call G is G66.1. a G hívás G66.1 típusú

If the value of the is 0, the macro will be called at the end of each motion block. If the value of the parameter 1, the macro will be called by each block.

Referring to the same code G in the body of the macro G

If a ***standard code G*** (for example G01) ***is designated as user call*** and it is referred again ***in the body of the macro***, this reference will not result in an additional macro call, but ***it will be interpreted*** and executed ***as a usual code G*** by the control.
If a ***non-standard code G*** (for example G101) ***is designated as user call*** and ***the calling code G*** (in this case G101) is referred again ***in the body of the macro***, ***the error message*** '2123 Not implemented function: G<number>' ***will be sent by the control***.

Referring to M, S ... in the body of a macro G, referring to G in the body of an M, S ... call

– Calling a system subprogram/ macro M, S, T, A, B, C, ASCII from calling a system macro G, and
– calling a system macro G from calling a system subprogram/ macro M, S, T, A, B, C, ASCII

is enabled depending on the following parameter state:
N1755 Macro Contr #1 ENC=0: call will not be initiated (they will be executed as usual code M, S, ...G);
N1755 Macro Contr #1 ENC=1: call will be initiated.

The argument set of the system macros G

The argument set can be defined as follows:
– if the type of the code is G65 or G66, it will be the argument set assigned to the G65, as well as P and L;
– if the type of the code is G66.1, the subject-matter mentioned at the code G66.1will apply to its argument set.

The modal call can be cancelled by instruction G67.

23.3.5 System Macro Call by Codes M Given in Parameter

For macro call, single codes M or arrays of codes M given in parameter can be assigned for each channel.

Assigning 10 single codes M for system macro call

In the parameter field, for each channel, maximum 10 different codes M can be assigned, for which macro call can be initiated. In this case, instead of the instruction line
Nn G65 Pp <argument assignment>
the instruction line
Nn **Mm** <argument assignment>
has to written. In this case, ***the code Mm will not be transferred to the PLC***, but the macro with appropriate program number will be called.
In the parameter field it is necessary to set which calling code M has to call which program number. These parameters are the following:
N1733 M(9020): the code M calling the program with the name O9020
N1734 M(9021): the code M calling the program with the name O9021
:
N1742 M(9029): the code M calling the program with the name O9029
If 0 is written in the parameter, the macro with the given program number will not be called.

Assigning an array of codes M for system macro call

By using the parameters below, ***an array of codes M can be assigned*** for macro call, for each channel.

In the parameter N1743 Start M Macro, the initial number of the array of codes M is specified by decimal integer number.

In the parameter N1744 Start Prg No. M Macro, the program number of the macro belonging to the code M given in the parameter Start M Macro is specified.

In the parameter N1745 No. of M Macro Codes , the number of elements of the array of codes G is specified.

If No. of M Macro Codes=0, for these codes M macro will not be called.

Example

Let, for example, the initial code of the array be M500. Therefore, Start M Macro=500 has to be set.
If the code M500 calls the program O3500, then No. M Macro=3500.
In the parameter No. of M Macro Codes , the number of codes M belonging to the array can be specified. If there is 20 codes in the array, then No. of M Macro Codes=20.
So, relation between the codes M and the program numbers is as follows:

Code M 1	M500	➜	Program number 1	O3500
Code M 2	M501	➜	Program number 2	O3501
Code M3	M502	➜	Program number 3	O3502
....				
Code M 20	M520	➜	Program number 20	O3519

Position of the code M starting a macro call in the block

In the block, the code M assigned in the parameter field and initiating a macro call can be preceded only by the symbol / and the address N (block number, the input argument can only follow then.

The macro called by code M is not modal.

By using code M, always G65-type, so ***non-modal*** call can be specified.

Referring to the same code M in the body of the macro M

If, ***in the body of the macro***, ***the same code M is referred*** again, the macro will not be called again, but ***the code M will be transferred to the PLC***.

Referring to G, S ... in the body of a macro M, referring to M in the body of an G, S ... call

– Calling a system subprogram/ macro G, S, T, A, B, C, M, ASCII from macro call initiated by a code M, and
– macro call initiated by a code M from calling a system subprogram/ macro G, S, T, A, B, C, ASCII

is enabled depending on the following parameter state:

N1755 Macro Contr #1 ENC=0: call will not be initiated (they will be executed as usual code M, S, ...G);

N1755 Macro Contr #1 ENC=1: call will be initiated.

The argument set of the codes M starting a macro call

In the block containing ***macro call started by code M***, ***both argument sets***, i.e. either the set 1 or the set 2, can be used.
Following the specification of the calling code M, the second code M will be interpreted as argument and transferred to the variable #13 by the control.

23.3.6 System Subprogram Call by the Code M Given in Parameter

For system subprogram call, single codes M or an array of codes M given in parameter can be assigned for each channel.
The difference between the code M calling a subprogram and the code M calling a macro is that ***the code M calling a subprogram does not transfer any argument to the subprogram***, but the code calling a macro does it.

Assigning 10 single codes M for system subprogram call

In the parameter field, for each channel, maximum 10 different codes M can be assigned, for which subprogram call can be initiated. In this case, ***the code Mm will not be transferred to the PLC***, but the macro with appropriate program number will be called, i.e. instead of the instruction

Nn Gg Xx Yy M98 Pp

the instruction

Nn Gg Xx Yy **Mm**

has to written..
In the parameter field it is necessary to set which calling code M has to call which program number. These parameters are the following:

N1720 M(9000): the code M calling the the program with the number O9000
N1721 M(9001): the code M calling the the program with the number O9001
:
N1729 M(9009): the code M calling the program with the number O9009

If 0 is written in the parameter, the subprogram with the given program number will not be called.

Assigning an array of codes M for system subprogram call

By using the parameters below, an array of codes M can be assigned for subprogram call, for each channel.

In the parameter N1730 Start M SubP, the initial number of the array of codes M is specified by decimal integer number.
In the parameter N1731 Start Prg No. M SubP , the program number of the subprogram belonging to the code M given in the parameter Start M SubP is specified.
In the parameter N1732 No. of M Codes, the number of elements of the array of codes M is specified.

If No. of M Codes=0, for these codes M subprogram will not be called.

An example:
Let, for example, the initial code of the array be M600. Therefore, Start M SubP=600 has to be set.
If the code M600 calls the subprogram O3600, then Start Prg No. M SubP=3600.
In the parameter No. of M Codes , the number of codes M belonging to the array can be specified. If there is 30 codes in the array, then No. of M Codes=30.
So, relation between the codes M and the program numbers is as follows:

Code M 1	M600	➜	Program number 1	O3600
Code M 2	M601	➜	Program number 2	O3601
Code M3	M602	➜	Program number 3	O3602
....				
Code M 20	M630	➜	Program number 20	O3629

Position of the code M starting a subprogram call in the block

In the block, the code M can be written in arbitrary position.

Referring to the same code M in the body of the macro M

If, ***in the subprogram***, ***the same code M is referred*** again, the subprogram will not be called again, but ***the code M will be transferred to the PLC***.

Referring to G, S ... in the body of a subprogram M, referring to M in the body of an G, S ... call

– Calling a system subprogram/ macro G, S, T, A, B, C, M, ASCII from subprogram call initiated by a code M, and
– subprogram call initiated by a code M from calling a system subprogram/ macro G, S, T, A, B, C, ASCII

is enabled depending on the following parameter state:

N1755 Macro Contr #1 ENC=0: call will not be initiated (they will be executed as usual code M, S, ...G);

N1755 Macro Contr #1 ENC=1: call will be initiated.

23.3.7 System Subprogram Call by Codes A, B, C, S, T Enabled in Parameter

Having been enabled, a subprogram can be called by codes A, B, C, S and T, for each channel.

Assigning codes A, B, C, S, T for subprogram call

At several bit positions of the parameter N1746 ABCST, the subprogram calls are as follows:

when #0 AM=1, the subprogram O9030.nct will be called by the code A;
when #1 BM=1, the subprogram O9031.nct will be called by the code B;
when #2 CM=1, the subprogram O9032.nct will be called by the code C;
when #3 SM=1, the subprogram O9033.nct will be called by the code S;
when #4 TM=1, the subprogram O9034.nct will be called by the code T.

In the case of addresses designated for calling subprograms, the values A, B, C, S and T will not be transferred either to the interpolator (if the address A, B or C is assigned to axis), or to the PLC, but the subprograms above will be initiated by the codes A, B, C, S and T.
If, for example, the code T is assigned to call a subprogram, the block

Gg Xx Yy **Tt**

will be equivalent to the following two blocks:

#199=t
Gg Xx Yy M98 P9034

Transferring the argument of the codes A, B, C, S and T to the subprogram

The values assigned to the addresses A, B, C, S and T, as arguments, will be written in the following ***global variables***:

Code A ➔ #195
Code B ➔ #196
Code C ➔ #197
Code S ➔ #198
Code T ➔ #199

Then, these variables can be used by the subprogram.

Position of the codes A, B, C, S and T starting a subprogram call in the block

In the block, the codes can be written in arbitrary position.

Referring to the calling code in the body of the subprogram A, B, C, S and T

If, in the subprogram being called to ***the address A, B, C, S and T***, the calling address is referred again, the subprogram will not be called again, but ***the code A, B, C, S and T will be transferred either to the interpolator or to the PLC.***

Referring to G, S ... in the body of a subprogram A, B, C, S and T, referring to A, B, C, S, T in the body of an G, S ... call

– Calling system subprogram/macro G, M, S, T, A, B, C, ASCII from subprogram call initiated by the code A, B, C, S, T (if the call initiated not from the calling address), and

– calling a subprogram initiated by the code A, B, C, S, T from system subprogram/macro G, M, S, T, A, B, C, ASCII call (if the call initiated not from the calling address)

is enabled depending on the following parameter state:

N1755 Macro Contr #1 ENC=0: call will not be initiated (they will be executed as usual code M, S, ...G);

N1755 Macro Contr #1 ENC=1: call will be initiated.

23.3.8 System Subprogram Call by Codes ASCII Given in Parameter

Subprogram can be called by four codes ASCII given in parameter, for each channel. ***The letters of the English alphabet*** can be selected from ***the codes ASCII.***

Assigning the code ASCII for subprogram call

In the parameters

N1747 ASCII Code SubP1
N1748 ASCII Code SubP2
N1749 ASCII Code SubP3
N1750 ASCII Code SubP4

4 different codes (letters of the English alphabet) can be set. Then, by these parameters, the control will call subprograms with the number (Onnnn) specified in the following parameters:

N1751 Prg No. ASCII Call1
N1752 Prg No. ASCII Call2
N1753 Prg No. ASCII Call3
N1754 Prg No. ASCII Call4

Transferring the argument of the code ASC;q4ubprogram

The values assigned to the address ASCII, as argument, will be written in the following ***global variables***:

Code ASCII 1 ➜ #191
Code ASCII 2 ➜ #192
Code ASCII 3 ➜ #193
Code ASCII 4 ➜ #194

Then, these variables can be used by the subprogram.

Position of the code ASCII starting a subprogram call in the block

In the block, the codes can be written in arbitrary position.

Referring to the calling code in the body of the subprogram being called by the code ASCII

If, ***in the subprogram being called by the code ASCII, the calling address is referred*** again, the subprogram will not be called again, but ***the code ASCII will be transferred either to the interpolator or to the PLC***.

Referring to G, S ... in the body of the subprogram ASCII, referring to ASCII in the body of an G, S ... call

– Calling system subprogram/macro G, M, S, T, A, B, C, ASCII from subprogram call initiated by the code ASCII (if the call initiated not from the calling address), and

– calling the subprogram initiated by the code ASCII from system subprogram/macro G, M, S, T, A, B, C, ASCII call (if the call initiated not from the calling address)

is enabled depending on the following parameter state:

N1755 Macro Contr #1 ENC=0: call will not be initiated (they will be executed as usual code M, S, ...G);

N1755 Macro Contr #1 ENC=1: call will be initiated.

23.3.9 Displaying the Blocks of Macros and Subprograms in Automatic Mode

Basically, the blocks of the macros and subprograms being under execution are listed by the control. However, it is possible to disable listing the blocks. In this case, the control will consider the whole macro or subprogram a block, and, in the block-by-block mode, it will not stop at the inner blocks.

Listing can be enabled or disabled by the bits #0 MD8 and #1 MD9 of the parameter N1756 List Contr. Listing the macros and subprograms numbered from 8000 to 8999 is controlled by the bit MD8, but listing those numbered from 9000 to 9999 is controlled by the bit MD9.
If the value of the bit MD8 or MD9 is 0, the blocks of the macro and the subprogram will not be listed during execution of the macros and the subprograms from 8000 to 8999 and from 9000 to 9999, respectively; execution will not stop at the inner blocks in the block-by-block mode.
If the value of the bit MD8 or MD9 is 1, the blocks of those macros and subprograms will also be listed, and the control will also stop at the inner blocks in the block-by-block mode.

23.4 Interruption-type Macro

In the course of running the part program, it is possible to suspend the actual program, to call another program, and then, after execution of this program, to continue the suspended one. The events causing interruption can be, for example, power failure, tool break or data gathering done between times.
The program being called during suspension is named ***interruption-type macro***.
Calling the interruption-type macro is started by the signal CP_MINT ***from the PLC program***.
In the part program, ***calling the interruption-type macro has to be enabled***, otherwise the PLC signal will be ineffective.
In addition, the program to be run due to the signal has to be designated.
The instruction

M96 P(program number)

or

M96 <filename>

enables the interruption signal CP_MINT from the PLC program ***to influence***. Due to the interruption signal, ***the program with the number given*** at the address P (program number) will be called.

The rules of programming the program number given at the address P are dealt with in the subsection **14.4.1 Identification of Programs in Memory. The Program Number (O)** on page 117. ***The rules of extension and in-memory location of the interruption-type macros given by the program number*** are the same as those of subprograms. See the subsection **14.4.2 Calling a Subprogram (M98)** on page 117.

Location of the interruption-type macro in the memory

The bit state #5 SYM of the parameter N1758 Intrrt Contr determines where in the memory the interruption-type macro has to be searched by the control:
– in the case of #5 SYM=0: always in the folder, in which the main program is there, even if the interruption-type macro was enabled not in the main program by the instruction M96P;
– in the case of #5 SYM=1: always in the directory of the folder SystemMacros, which directory belongs to the appropriate channel.

Return from the interruption-type macro occurs ***by the code M99***.

The instruction

M97

disables calling the interruption-type macro, i.e. the interruption-type macro will not be called any more even if the signal is received. Interruption will also be disabled by reset and program end.

The interruption-type macro can exclusively be used

in ***automatic*** or ***manual data input*** mode,

when ***start*** state is effective.

Enabling the function to work

In the control, the bit state #0 USD=1 of the ***parameter*** N1758 Intrrt Contr ***enables the interruption-type macro to work***.
If the value of the parameter is #0 USD=0, the interruption-type macro will not work, and the PLC can use the codes M96 and M97 as function.

Enabling interruption by other codes

In the bit state #4 MCD=0 of the parameter N1758 Intrrt Contr, ***enabling/disabling*** the interruption-type macro can be done by using the code pair ***M96/M97***.
In the bit state #4 MCD=1 of the parameter N1758 Intrrt Contr, enabling/disabling the interruption-type macro can be done by using a code pair different from the M96/M97.
In this case, ***the code M enabling the interruption*** can be given in the ***parameter*** N1759 M Code MI On, but ***the code M disabling the interruption*** can be given in the parameter N1760 M Code MI Off.
It can be useful in the case, when the code M96 or M97 is already used by the PLC for other function.

Subprogram-type or macro-type interruption

In the bit state #1 STP=1 of the parameter N1758 Intrrt Contr, interruption will be ***subprogram-type***, i.e. the level of local variables will not be higher by the call, the values of the local variables in the called program and in the interrupted program will be the same.
On the other hand, in the bit state #1 STP=0, interruption will be ***macro-type***, the level of local variables will be higher in the called program, i.e. the values of the local variables in the calling program and in the interrupted program will be different.

Edge-controlled and level-controlled interruption

Interruption is ***edge-controlled*** in the case, when the interruption-type macro is called by the ***rising edge*** of the PLC signal CP_MINT. If the signal remains switched on after returning from the interruption-type macro, the macro will be called again only after switching the signal off and then switching it on again.
On the contrary, in the case of ***level-controlled*** interruption, if the signal CP_MINT remains switched on after returning from the interruption-type macro, the interruption-type macro ***will be called again*** until the signal CP_MINT will be switched off.

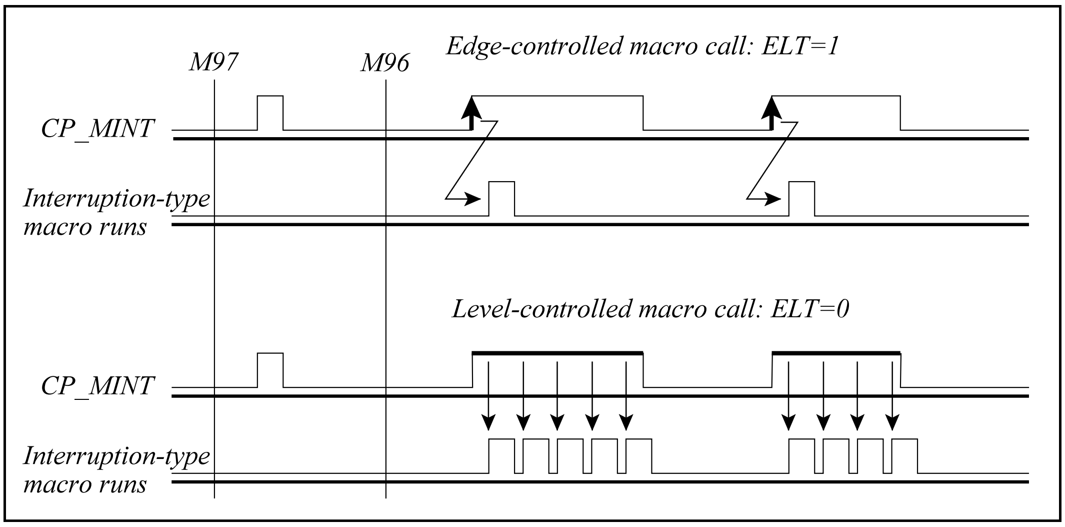

M97
M96
Edge-controlled macro call: ELT=1
CP_MINT
Interruption-type macro runs
Level-controlled macro call: ELT=0
CP_MINT
Interruption-type macro runs

Fig. 23.4-1

The bit #3 ELT of the parameter N1758 Intrrt Contr determines the type of the macro call. If #3 ELT=1, the macro call will be edge-controlled, if #3 ELT=0, the macro call will be level-controlled.

Interruption during block execution or at the end of the block

The user can decide whether the interruption-type macro begins to run ***at the moment of receiving the signal***, or it waits until motion will be completed in the actual block and begins to run ***only at the end of the block***.
In the bit state #2 TPI=0 of the parameter N1758 Intrrt Contr , the interruption-type macro will be called immediatel, but in the bit state #2 TPI=1 it will be called only at the end of the block.

Modal information in the interruption-type macro

Calling the interruption-type macro differs from a normal subprogram call.
In the case of a subprogram call (M98Pp), the subprogram inherits from the calling program
the modal information#4001...#4130 valid in the previous block, and
the modal information#4201...#4330 valid in the block being executed.
Certainly, these two kinds of information are different, because, due to buffering, execution lags behind preprocessing the blocks.
In the case of the interruption-type macro, the situation is different.
Interruption occurs during execution of the program, therefore ***the interruption-type macro (M96Pp) can read out the modal information valid in the block being executed, i.e. in the interrupted block, in the variables #4401...#4530***.
The modal information #4001...#4130 and #4201...#4330 will be initialized after entering the interruption-type macro.
During execution of the interruption-type macro, the variables #4001...#4130 and #4201...#4330 work in the usual way, but the variables #4401...#4530 retain the modal information valid at the moment of interruption.

Return from the interruption-type macro by M99

The control ***stores the modal information valid at the moment of interruption***, and then, ***after return by M99, it restores the saved modal information***.
If the program is interrupted during interpolation, the program will finish interpolation after return; if the program is interrupted at the end of a block, the succeeding block will be taken and executed.
Therefore, if motion has to also be programmed in the interruption-type macro, it will be advisable to store the block-end position (#5001 ...) after entering the interruption-type macro, and then, prior to return, to move back to this position.

Return from the interruption-type macro by M99Pp

If return from the interruption-type macro is executed by the instruction M99Pp, the machining will be continued from that block of the interrupted program the number of which is given at the address P.
After return by the M99Pp, the saved modal information will not be restored by the control, but the program will be continued using the modal information generated in the interruption-type macro.

23.5 NC and Macro Instructions. Execution of Macro Blocks

NC and macro blocks

In the program language, ***NC and macro blocks*** can be distinguished.
The blocks written with traditional codes G, M etc. are considered to be ***NC blocks***, even if the values of the addresses are not only numbers but variables or formula as well.
The following blocks are considered to be ***macro instructions***:
– the blocks containing assignment statement: #i=#j;
– the blocks containing conditional or cycle organizing instruction: IF, WHILE;
– the blocks containing control instructions: GOTO, DO, END;
– the blocks containing macro call: G65, G66, G66.1, G67 or those codes G or M initiating macro call;
– the subprogram call (M98P or the subprogram initiated by the A, B, C, S, T and M);
– the code of return from a subprogram or a macro (M99).

Synchronization of instructions (G53)

In the bit state #6 MBM=1 of the parameter N1301 DefaultG2 (multibuffer mode), the block preparator reads both the NC and macro blocks ahead, processes them, and then, places them in the buffer memory. The executors, the interpolator and the PLC, take the blocks from the buffer memory during execution of the program. It is necessary in order that the interpolator will be able to move the axes continuously, and will not have to wait for processing the succeeding block.
As a result of block processing in advance and buffering, the executor lags behind the block preparator by hundreds of blocks. For example, the block preparator is already processing the block 1500 while the executor is executing the block 1000 yet.
This is the reason of distinguishing the variables #4001...#4130 (#_BUFx) and the variables #4201...#4330 (#_ACTx). While the former provides information about the blocks placed in the buffer memory last, the latter gives information about the blocks being executed.
In some cases, preprocessing and buffering the blocks has to be suspended; for example, when it is necessary to wait for a given position of an axis during program execution.
The instruction

G53

written in a separate block suspends reading the blocks ahead. The control waits until the block buffer will be empty and only after that begins to read in and process the succeeding block.

Execution of macro blocks

The control can execute the macro blocks in parallel with execution of NC blocks or following the execution of them. It is the parameter N1755 Macro Contr #2 SBM that controls execution of NC and macro blocks. If the value of the parameter is:

=0: the NC and macro blocks will be executed in the sequence written in the program;
=1: macro instructions will be executed during execution of the NC blocks.

Example

SBM=0

```
%O1000
...
N10 #100=50
N20 #101=100
N30 G1 X#100 Y#101
N40 #100=60 (assignment statement
after N30)
N50 #101=120 (assignment statement
after N30)
N60 G1 X#100 Y#101
```

Assignment statement written in the blocks N40 and N50 will be carried out after execution of the block N30.

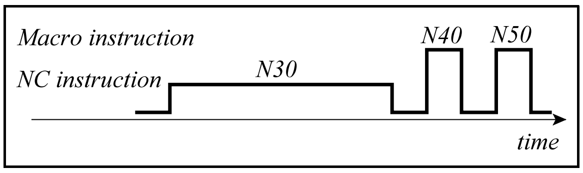


Fig. 23.5-1

☞ *Consequences:*
– slower program execution,
– if execution of the block N30 is interrupted and then machining is started again, and since the variables of the block N30 are not rewritten by the blocks N40 and N50 yet, machining can be continued in a usual way.

SBM=1

```
%O1000
...
N10 #100=50
N20 #101=100
N30 G1 X#100 Y#101
N40 #100=60 (assignment statement
during N30)
N50 #101=120 (assignment statement
during N30)
N60 G1 X#100 Y#101
```

Assignment statement written in the blocks N40 and N50 will be carried out during execution of the block N30.

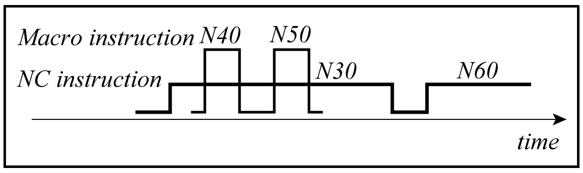


Fig. 357-2

☞ *Consequences:*
– faster program execution,
– if execution of the block N30 is interrupted and then machining is started again, and since the variables of the block N30 are rewritten by the blocks N40 and N50 already, machining can be continued only in the case, when search is started for the block N30.

23.6 Pocket Milling Macro Cycle

The instruction

G65 P9999 X Y Z I J K R F D E Q M S T

starts a pocket milling cycle.

Prior to calling the cycle, the tool must be positioned over the geometric center of the pocket in the selected plane, at a safety distance from the workpiece surface. At the end of the cycle the tool will be positioned back to the same point.

Interpretation of the addresses of the block is as follows:

X: the size of the pocket in the direction X;

Y: the size of the pocket in the direction Y;

Z: the size of the pocket in the direction Z.

It is determined by the instructions G17, G18 and G19, which of the three coordinates will be the length, width and depth of the pocket. For example, in the case of G17, Z will be the depth of the pocket, the longer one of X and Y will be the length of the pocket and the shorter one will be the width thereof.

These values have to be entered in absolute values as positive numbers.

R: the radius of the corners of the pocket.

Rounding (if any) of the pocket corners has to be specified at the address R. If the address R is not filled, the rounding of the pocket corners will be equal to the tool radius.

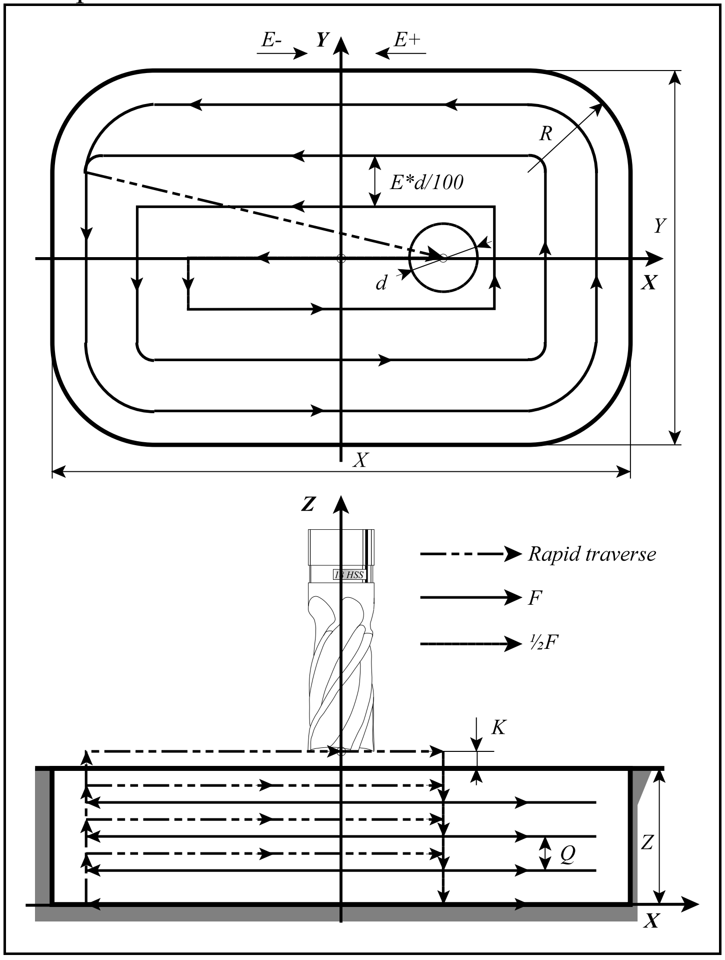


Fig. 23.6-1

I: the safety distance in the direction of depth in the case of G19.

J: the safety distance in the direction of depth in the case of G18.

K: the safety distance in the direction of depth in the case of G17.

Depending on the plane selected, the safety allowance in the direction of the tool has to be specified at the addresses I (G19), J (G18) or K (G17) in the block. When the cycle is started, it is assumed by the control, that the tip of the tool is located at such a distance from the workpiece surface. In the course of milling the pocket, when removing a level of material is completed, the tool will be retracted by such a distance so that it can be positioned to the start point for milling the next level.

D: the address of the cell containing the tool radius compensation.

The radius compensation cell number of the tool used in the program has to be specified mandatorily at the address D. Otherwise, milling a pocket has to be carried out in the state

G40.
E: the width of cutting in percent of the milling tool diameter:

with the sign +: counter clockwise machining;
with the sign −: clockwise machining.

At the address E, the types of information can be given to the control. The value of E determines how great the width of cutting has to be in percent of the milling tool diameter. If it is not given, its value is automatically assumed by the control to be +83%. Depending on the width of the pocket, the data given at the address E can be specified by the control in such a way so that the value of infeed will be constant in the course of milling a level. However, modification has to be decrease only. The sign of the address E shows the direction of milling. In the case of E+, i.e. when it is positive, machining is being performed in the counter clockwise direction; in the case of E−, i.e. when it is negative, machining is being performed in the clockwise direction.
Q: depth of cut
At the address Q, the depth of cut can be given in the unit system used, i.e. in mm or in inch. Depending on the depth of the pocket, the programmed value can be revoked by the control in order to obtain constant division of cut. However, modification has to be decrease only.
F: feed
At the address F, the value of the feed used in the course of cycle can be given. If no value is given at the address F, the modal value of F will be taken into account by the control. 50% of the value of F will be applied in the following cases:
– When machining a level is started and drilling to the depth Q is executed in the direction of the tool.
– During longitudinal milling, as long as the tool is loaded on both sides.
M S T: function
In the block calling the pocket milling, one function M and functions S and T can be given, which will be executed prior to starting the milling.

Degenerate cases of pocket milling

If the width of the pocket is not given, the radius of the pocket corners will be taken on twice and it will be the width of the pocket.

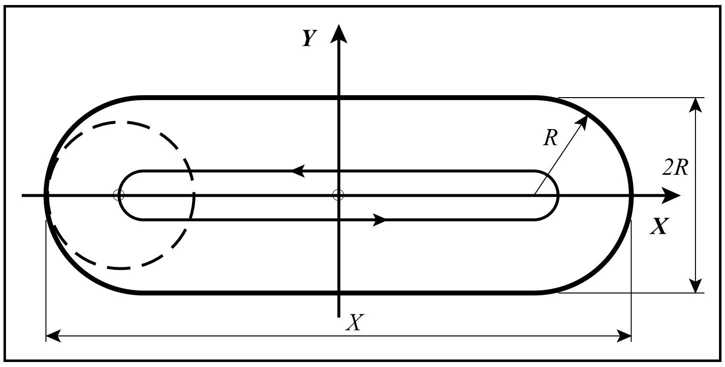


Fig. 22.6-2

If neither the width of the pocket nor the radius of the corner is given, the diameter of the tool applied will be taken as the with of the pocket (groove).

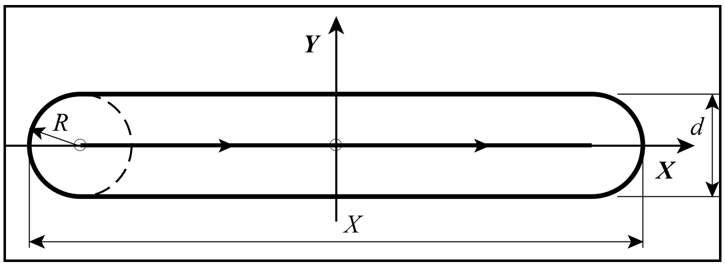


Fig. 23.6-3

If neither the length nor the width of the pocket is given, and only the address R is programmed, a circular pocket will be milled.

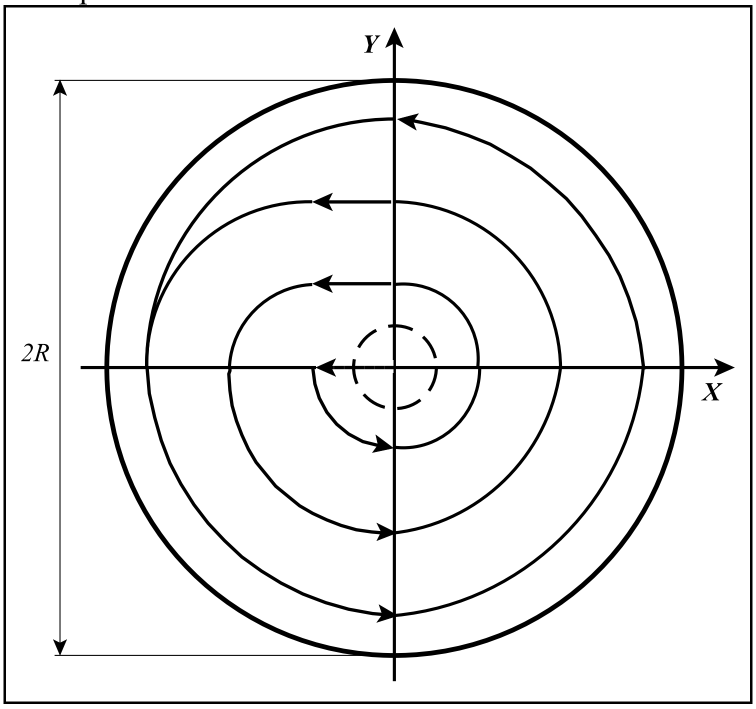


Fig. 23.6-4

If neither the length nor the with and the radius is given, the cycle will degenerate into drilling.

Error messages which are possible during execution of pocket milling:

MACRO ERROR 1: wrong block specification. Possible causes:

– The depth of the pocket is not specified.
– The tool radius is not specified.
– The depth of cut is not specified.

MACRO ERROR 2: wrong dimensional specification. Possible causes:

– The dimension given as the length or the width of the pocket smaller than the double of the pocket radius.
– The value of the length and the width of the pocket smaller than tool diameter called at the address D.

– The value of the width of cutting is 0, or the tool diameter called is 0.
– The value of the depth of cut is 0, i.e. 0 is programmed at the address Q.

24 Writing and Reading the Parameters

The parameters are used to set up operation of the control according to the demands. The parameters are stored in the non-volatile memory of the control. The parameters of the control can be rewritten and read in from the part program.
Each of the parameters has an ***identifier number*** of maximum ***four digits***, which identifies the parameter in the memory. ***One value*** (in the case of global parameters) or ***several different values*** (indexed per channel, per axis, per spindle) ***can belong to these identifier numbers***.
The parameters can be classified in accordance with ***number representation***, ***effect range*** and ***case of taking into account***.

Classification of parameters in accordance with number representation

Bit-type parameters: their value range can be
0 or 1.
Integer-type parameters (DWORD): their value range can be
without sign: 0,...4294967295 or
with sign: -2147483648... +2147483647.
Floating point-type parameters (double): their value range can be
in the case of negative numbers: -1.7×10^{308} ... -5.0×10^{-324}
in the case of positive: 5.0×10^{-324} ... 1.7×10^{308}.

Classification of parameters in accordance with effect range

Global parameters: Those parameters are named global which are valid in every channel of the control. For example, the parameters N2201 Waiting M Codes Min and N2202 Waiting Codes Max are such ones, which designate the M codes group controlling synchronization between channels and are valid for each of the channels.
In the case of global parameters, one parameter value belongs to each identifier number.
Parameters that can be specified per channel: These are the parameters, in which values different per channel can be written. For example, the group N1500 Return Val G73 is such a parameter, in which the distance of retraction can be set up in the drilling cycle G73 and values different per cannel can be written.
In the case of parameters that can be specified per channel, ***maximum 8*** parameter values can belong to each identifier number.
Parameters that can be specified per axis: These are the parameters, in which values different per axis can be written. For example, the bit #0 DIA of the group N0106 Axis properties is such a parameter, in which it can be set per axis whether the the given axis shoukd be programmed in radius or in diameter.
In the case of parameters that can be specified per axis, ***maximum 32*** parameter values can belong to each identifier number.
Parameters that can be specified per spindle: These are the parameters, in which values different per spindle can be written. For example, the group N0608 Spindle Encoder Counts is such a parameter, in which the number of pulses of the encoder mounted on the given axis can be specified per spindle.
In the case of parameters that can be specified per spindle, ***maximum 16*** parameter values can belong to each identifier number.

Classification of parameters in accordance with case of taking into account

Parameters taken into account during run-time: These parameters will be taken into account by the control immediately after rewriting them.

Parameters taken into account after restart: Rewriting these parameters will be taken into account by the control only after restart (after turn-off and then turn-on).

Parameters that can be rewritten in the case of emergency stop state: Rewriting these parameters is enabled by the control only in the case of emergency stop state.

Detailed description of the parameters can be found in the manual 'NCT2xx Machine Tool Controls Parameters'.

24.1 Writing the Parameters from Part Program (G10 L52)

The command

G10 L52 (writing the parameter on)

written in separate row turns on the function of writing the parameter. Then, the serial number and the value of each parameter has to be specified in separate row in accordance with the following way:

N_ (Q_) R_ (writing global parameter)
N_ P_ (Q_) R_ (writing channel/axis/spindle parameter)
...
...

It is the command

G11 (end of writing the parameter)

written in separate row that closes the end of writing the parameter.
Interpretation of the addresses N, P, Q and R:

N: the identifier number of the parameter (0-9999). The leading zeros can be neglected.

☞ ***Warning!*** *Only such parameter having a given number can be rewritten by the use of the command G10 L52, which does not require restart for being taken into account or does not require emergency stop state for being rewritten!*

P: the number of the channel (1-8), the axis (1-32) or the spindle (1-16). The value of P that can be assigned in the specific cases are given in brackets.

☞ ***Warning!*** *If a parameter, which can be specified per channel, is being written and the address P is not filled in, the command will rewrite the parameter of the channel, in which the program is running.*

Q: in the case of the bit-type parameter, it is the number of the bit to be written from 0 to 7.

R: the value of the parameter. For the value of R, the incremental operator I is accepted; it increases the actual value by the value specified at the address RI. In the case of floating-point data, decimal point (.) can be used.

Parameters modified by the command G10 L52 will be saved so modification will be valid during next turn-on too.

If the addresses N, P, Q or R addresses are filled in incorrectly, the control will send the message ‘2002 <N, P, Q or R> data out of range‘

to the address N: if there is no parameter having such identifier number;
to the address P: if there is no channel/axis/spindle having such number;
to the address Q: if the value of Q is less than 0 or greater than 7;
to the address R: if a value being out of the value range permitted for the parameter having identifier N is written on R.

If any of the data N, P, Q and R is missing, the control will send the error message ‘2004 <N, P, Q, R> data missing’.
If such parameter should be rewritten, which requires restart for being taken into account or requires emergency stop state for being rewritten, the control will send the error message ‘2193The parameter Nnnnn cannot be modified by the command G10’.
An example of writing the parameters:

```
...
G10 L52             (Writing the parameters on)
N107 P4 Q0 R1       (N0107 RollOver Control A4- REN)
N1339 (P1) R0.5     (N1339 Radius Diff)
N1746 (P1) Q1 R1    (N1746 ABCST BM=1)
G11                 (Writing the parameters off)
...
```

24.2 Reading the Parameters from Part Program (PRM)

Using the assignment statements

#a=PRM[#b,#c]

and

#a=PRM[#b,#c] / [#d]

the value of ***any parameter*** of the control ***can be read out without any constraint***. The meaning of the arguments is as follows:

#a: it is a writable macro variable.
#b: it is the identifier number of the parameter. It can be given indirectly on a macro variable or directly using a number value.
#c: it is the bit number of the parameter in the case of reading a bit-type parameter. It can be given indirectly on a macro variable or directly using a number value.
#d: it is the number of the channel (1-8), the axis (1-32) or the spindle (1-16). The value that can be assigned on #d in the specific cases are given in brackets. It can be given indirectly on a macro variable or directly using a number value.

☞ ***Warning!***

*If a parameter which can be specified per channel is being read and the part / **[#d]** is neglected from the command, the command will read in the parameter of the channel, in which the program is running.*
If the arguments of the command PRM are specified on macro variables and the value of the macro variable is #0 (empty), it would produce such an effect as if that argument would not have been specified.
Arguments could also be results of macro expressions.

If the command PRM stands on the right side of a command, which is not an assignment statement, the control will send the error message ‘2017 Illegal address PRM’.

If the argument #b of the command PRM, i.e. the number of the parameter id missing, the control will send the error message ‘2064 Syntax error’.
If the bit number #c or the argument /[#d] of the command PRM is missing, the control will send the error message ‘2194 PRM function param <2., 3.> missing’.

An example of reading the parameters:

```
...
#101=PRM[107,0]/[4]      (N0107 RollOver Control A4- REN)
#102=PRM[1339](/[1]      (N1339 Radius Diff)
#103=PRM[1746,1](/[1])   (N1746 ABCST BM=1)
#104=PRM[2201]           (N2201 Waiting M Codes Min)
...
```

Notes