# NCT®
# 201T

**Controls for Lathes**

# Programmer's Manual

Manufactured by **NCT Automation kft**.
H1148 Budapest Fogarasi út 7
⊠ Address: H1631 Bp. pf.: 26
☎ Phone: (+36 1) 467 63 00
☎ Fax:(+36 1) 467 63 09
E-mail: nct@nct.hu
Home Page: www.nct.hu

# Contents

February 6, 2012

# 1 Introduction

## 1.1 The 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.

Address Chain:

| Address | Meaning | Value limits |
|---|---|---|
| O | program number | 0001 - 9999 |
| / | optional block | 1 - 9 |
| N | block number | 1 - 99999 |
| G | preparatory function | * |
| X, Y, Z, U, V ,W | length coordinates | I, -, * |
| A, B, C, H | angular coordinates, auxiliary functions | I, -, * |
| R | circle radius, auxiliary data | I, -, * |
| I, J, K | circle center coordinates, auxiliary coordinate | -, * |
| E | auxiliary coordinate | -, * |
| F | feed rate | * |
| S | spindle speed | * |
| M | miscellaneous function | 1 - 999 |
| T | tool number | 1 - 9999 |
| L | repetition number | 1 - 9999 |
| P | auxiliary data, dwell time | -, * |
| Q | auxiliary data | -, * |
| ,C | distance of chamfer | -, * |
| ,R | radius of fillet | -, * |
| ,A | angle of straight line | -, * |
| ( | comment | * |

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, respectively.
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. To distinguish the end of block from the beginning of another block on the screen, each new block begins in a new line, with a character > placed in front of it, in the case of a block longer than a line, the words in each new line are begun with an indent of one character.

Program Number and Program Name

The program number and the program name are used for the identification of a program. The use of program number is mandatory that of a program name is not.
The address of a **program number** is O. It must be followed by exactly **four digits**.
The **program name** is any arbitrary character sequence (string) put between opening **"("** and closing brackets **")"**. It may have max. 16 characters.
The program number and the program name are separated by characters ␊ (**L**ine **F**eed) from the other program blocks in the memory.
In the course of editing, the program number and the program name will be displayed invariably in the first line.
There may be not two programs of a given program number in the backing store.

Beginning of Program, End of Program

Each program begins and ends with characters %. In the course of part program editing the program-terminating character is placed invariably behind the last block in order to ensure that the terminated locks will be preserved even in the event of a power failure during editing.

Program Format in the Memory

The program stored in the memory is a set of ASCII characters.
The format of the program is

```
%O1234(PROGRAM NAME)␊/1N12345G1X0Y...␊G2Z5...␊....␊
...␊
...␊
N1G40...M2␊
%
```

In the above sequence of characters,

| | |
|---|---|
| ␊ | is character "**L**ine **F**eed", |
| % | is the beginning (and end) of the program. |

Program Format in Communications with External Devices

The above program is applicable also in communications with an external device.

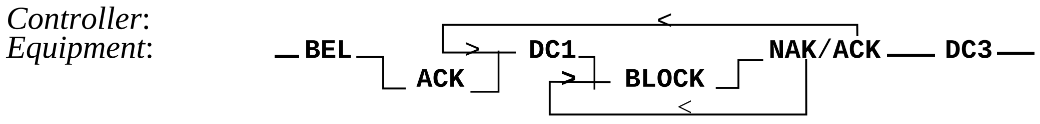
Main Program and Subprogram

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 have to be machined at different places, it is not necessary to write those program-sections over and over again in the main program, instead, a subprogram has 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.

DNC Channel

A program contained in an external unit (e.g., in a computer) can also be executed without storing it in the control's memory. Now the control will read the program, instead of the memory, from the external data medium through the RS232C interface. That link is referred to as "DNC channel". This method is particularly useful for the execution of programs too large to be contained in the control's memory.

The DNC channel is a protocol-controlled data transfer channel as shown below.



The above mnemonics have the following meanings (and their ASCII codes):

**BEL** (7): The control requests the sender to establish the communication. The control issues L again unless ACK is returned in a definite length of time.

**ACK** (6): Acknowledgment.

**NAK** (21): Spurious data transfer (e.g. hardware trouble in the line or BCC error). The transfer of BLOCK has to be repeated.

**DC1** (17): Transfer of the next BLOCK has to be started.

**DC3** (19): Interruption of communication.

**BLOCK**:

– Basically an NC block (including the terminating character ␊) and the checksum thereof (BCC) stored in 7 bits as the last byte of the block (bit 7, the uppermost one, of BCC is invariably 0). No SPACE (32) or some other character of lower ASCII code may be contained in the block.

– **EOF** (26) (**E**nd **O**f **F**ile), a signal is transferred by the Equipment ("sender") to interrupt the communication.

For the DNC mode, set the second physical channel (only that one is applicable as a DNC channel) for 8-bit even-parity mode.

A main program executed from the DNC channel may have a linear sequence only. This does not apply to subprogram or macro (if any have been called) however, they must be contained in the memory of control. In the event of a departure from the linear sequence in the main program (GOTO, DO WHILE), the control will return error message *3058 NOT IN DNC*. If the control detects a BLOCK error and returns NAK, the BLOCK has to be repeated.

## 1.2 Fundamental Terms

The 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 Z__
X__ Z__
```

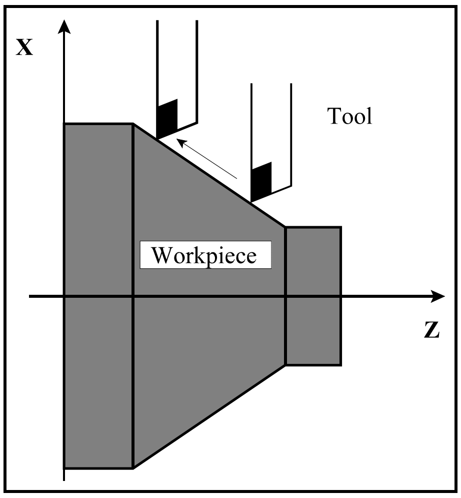


Fig. 1.2-1

Tool movement along an arc:

Program:
```
G02 X__ Z__ R__
```

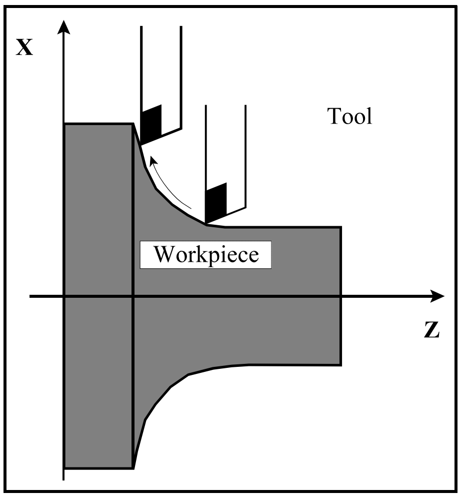


Fig. 1.2-2

Preparatory Functions (G codes)

The type of activity to be performed by a block is described with the use of preparatory functions (also referred to as G codes). 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.

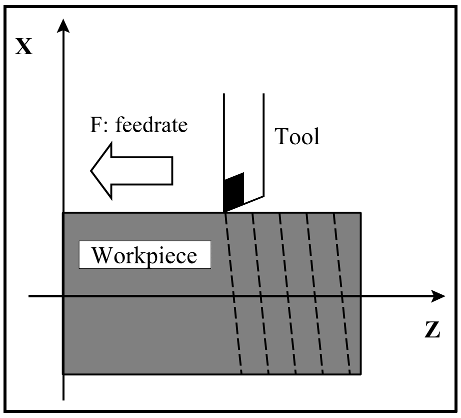


Fig. 1.2-3

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 have to be written at the coordinate address in the part program. E.g., X150 Z–100 means a coordinate point of 340 and –100 mm in the coordinate system of the workpiece in the direction X and Z respectively.
The coordinate system in which the control interprets the positions, is different from the coordinate system of the workpiece. For the control system to make a correct workpiece, the zero point offsets of the two coordinate systems have to be set. This can be achieved, e.g., by moving the tool tip to a point of a known position of the part and setting the coordinate system of the control to that value.

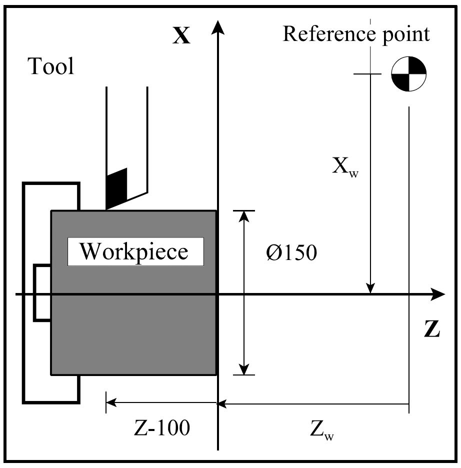


Fig. **1.2-4**

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 block

```
G90 X200 Z150
```

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

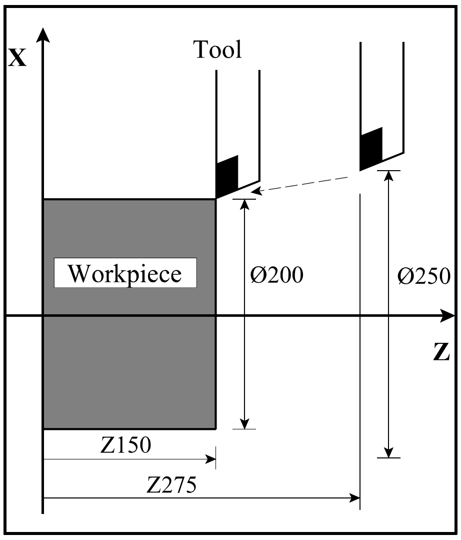


Fig. **1.2-5**

## 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:

```
        U–50 W–125
```

The code of incremental data specification is G91. Code G91 refers to all coordinate values.
The specification above is equivalent to the block below:

```
        G91 X–50 Z-125
```

It will move the tool over the above distance from its previous position.

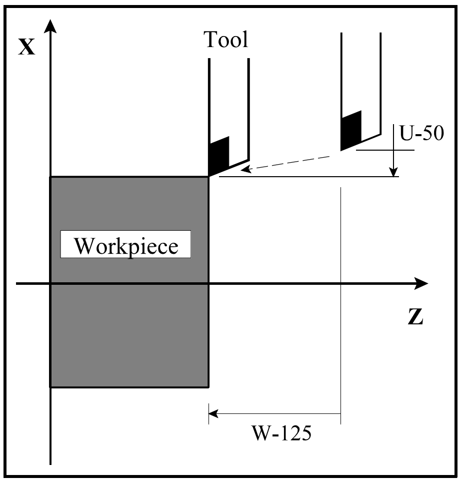


Fig. 1.2-6

## Diameter Programming

The coordinate X may be specified both in diameter or in radius depending on parameter.

## Modal Functions

Some codes are effective until another code or value is specified. These are **modal codes**. E.g., in program detail

```
        N15 G90 G1 X20 Z30 F0.2
        N16 X30
        N17 Z100
```

the code of G90 (absolute data specification) and the value of F (Feed), specified in block N15, will be modal in blocks N16 and N17. Thus it is not necessary to specify those functions in each block followed.

## One-shot (Non-modal) Functions

Some codes or values are effective only in the block in which they are specified. These are one-shot functions.

## Spindle Speed Command

The spindle speed can be specified at address S. It is also termed as "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 according to the diameter machined the way that the speed of the tool tip relative to the surface of the workpiece to be constant. This function is the constant surface speed control.

## Tool Function

In the course of machining different tools have to be employed for the various cutting operations. The tools are differentiated by numbers. Reference can be made to the tools with code T. The first two digits of the T code refer to the tool number (that is in which position in turret it can be found), while the second two digits refer to the code of offset compensation. Instruction

```
        T0212
```

in the program means that tool No. 2 has to be changed and the offset compensation group No. 12 is to be applied.

Miscellaneous Functions

A number of switching operations have to be carried out in the course of machining. For example, starting the spindle, turning on the coolant. Those operations can be performed with M (miscellaneous) functions. E.g., in the series of instructions

```
M3 M8
```

M3 means “rotate the spindle clockwise”, M8 means "turn on the coolant".

Tool Length Compensation

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

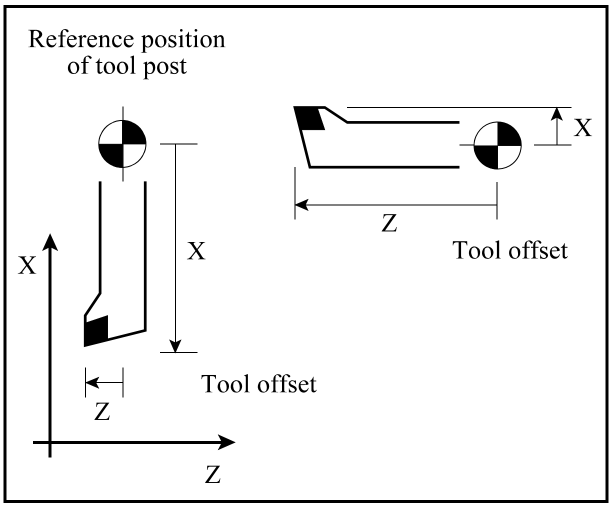


Fig. **1.2-7**

Tool Nose Radius Compensation

When machining a workpiece and the tool does not move parallel to one of the axes exact size can be achieved only if not the tool tip is moved on the programmed path but the tool nose center parallel to it and with the distance of r. Radius compensation has to be introduced in order to write the actual contour data of the part in the program, instead of the path covered by the tool tip . The values of radius compensations have to be set in control system. Hereinafter reference can be made to tool nose radius compensations at address T in the program.

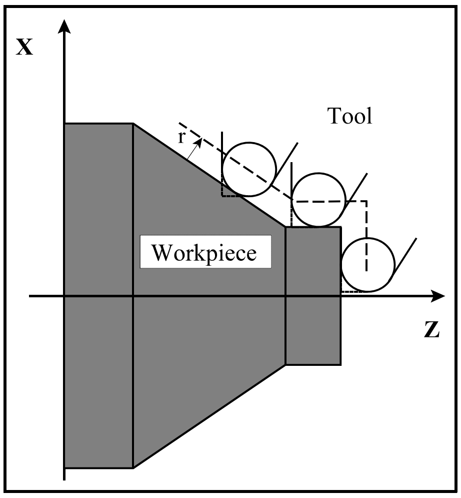


Fig. **1.2-8**

# 2 Controlled Axes

| | |
|---|---|
| Number of Axes (in basic configuration) | 2 axes |
| In expanded configuration | 6 additional axes (8 axes altogether) |
| Number of axes to be moved simultaneously | 8 axes (with linear interpolation) |

## 2.1 Names of Axes

The names of controlled axes can be defined in the parameter memory. Each address can be assigned to one of the physical axes.
In the basic configuration, the names of axes: X and Z.
The names of additional (expansion) axes depend on their respective types.
Possible names of expansion axes performing linear motions are: Y, U, V and W. When U, V, W axes are parallel to the main axes X,Y and Z, their name will be U,V and W, respectively.
Axes performing rotational motions are termed A, B and C. The rotational axes whose axle of rotation parallel to X, Y and Z directions are termed A, B and C, respectively.
The name of the spindle axis in case of polar or cylindrical coordinate interpolation is used: C.
In case U, V or W axis cannot be found in the machine at the above addresses incremental movements can be specified for the axes X, Y, Z respectively. Address H can be used for specifying incremental movement for C.

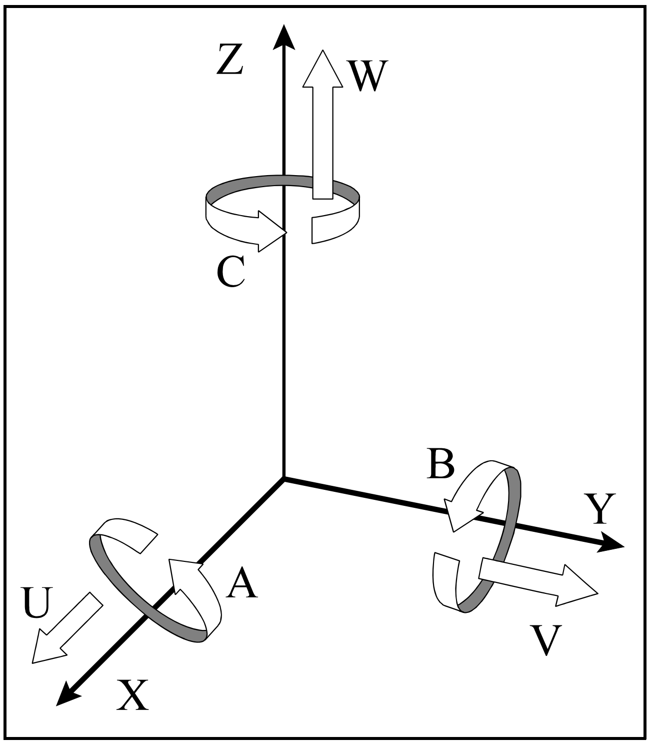


Fig. **2.1-1**

## 2.2 Unit and Increment System of Axes

The coordinate data can be specified in 8 digits. They can have signs, too. The positive sign + is omitted.
The data of input length coordinates can be specified in mm or inches. They are the units of input measures. The desired one can be selected from the program.
The path-measuring device provided on the machine can measure the position in mm or in inches. It will determine the output unit of measures, which has to be specified by the control system as a parameter. The two units of measures may not be combined among axes on a given machine.
In the case of different input and output units of measures, the control system will automatically perform the conversion.

The rotational axes are always provided with degrees as units of measure.
The **input increment system** of the control is regarded as the smallest unit to be entered. It can be selected as parameter. There are three increment systems available IS-A, IS-B and IS-C. The increment systems may not be combined for the axes on a given machine.
Having processed the input data, the control system will provide new path data for moving the axes. Their resolution is always twice the particular input increment system. It is termed the **output increment system** of the control.
Thus the input increment system of the control is determined by the resolution of the encoder.

| Increment system | Min. unit to be entered | Max. unit to be entered |
|---|---|---|
| IS-A | 0.01 mm | 999999.99 mm |
| | 0.001 inch | 99999.999 inch |
| | 0.01 degree | 999999.99 degree |
| IS-B | 0.001 mm | 99999.999 mm |
| | 0.0001 inch | 9999.9999 inch |
| | 0.001 degree | 99999.999 degree |
| IS-C | 0.0001 mm | 9999.9999 mm |
| | 0.00001 inch | 999.99999 inch |
| | 0.0001 degree | 9999.9999 degree |

Coordinate data of X axis can also be interpreted by the control in diameter, provided parameter 4762 DIAM is 1. In this case the value limits defined in the above table are interpreted in diameter with their value remaining so.

# 3 Preparatory Functions (G codes)

The type of command in the given block will be determined by address G and the number following it.
The Table below contains the G codes interpreted by the control system, the groups and functions thereof.

| G code | Group | Function | Page |
|---|---|---|---|
| G00* | 01 | Positioning | 21 |
| G01* | | Linear interpolation | 21 |
| G02 | | Circular interpolation, clockwise (CW) | 22 |
| G03 | | Circular interpolation, counter-clockwise (CCW) | 22 |
| G04 | 00 | Dwell | 52 |
| G05.1 | | Multi-buffer mode on | |
| G07.1 | | Cylindrical interpolation | 33 |
| G09 | | Exact stop (in the given block) | 46 |
| G10 | | Data setting (programmed) | 59, 77, 81 |
| G11 | | Programmed data setting cancel | |
| G12.1 | 26 | Polar coordinate interpolation on | 29 |
| G13.1 | | Polarc coordinate interpolation off | 29 |
| G17* | 02 | Selection of $X_pY_p$ plane | 62 |
| G18* | | Selection of $Z_pX_p$ plane | 62 |
| G19 | | Selection of $Y_pZ_p$ plane | 62 |
| G20 | 06 | Inch input | 35 |
| G21 | | Metric input | 35 |
| G22* | 04 | Programable stroke check function on | 184 |
| G23 | | Programable stroke check function off | 184 |
| G25* | 08 | Spindle speed fluctuation detection off | 67 |
| G26 | | Spindle speed fluctuation detection on | 67 |
| G28 | 00 | Programmed reference-point return | 53 |
| G29 | | Return from reference point | 54 |
| G30 | | Return to the 1st, 2nd, 3rd and 4th reference point | 54 |
| G31 | | Skip function | 181 |
| G33 | 01 | Thread cutting | 27 |
| G34 | | Variable-lead thread cutting | 28 |
| G36 | 00 | Automatic tool-length measurement X | 182 |
| G37 | | Automatic tool-length measurement Z | 182 |
| G38 | | Tool nose radius compensation vector hold | 99 |
| G39 | | Tool nose radius compensation corner arc | 99 |

| G code | Group | Function | Page |
|---|---|---|---|
| G40* | 07 | Tool nose radius compensation cancel | 83 |
| G41 | | Tool nose radius compensation left | 83, 83, 87 |
| G42 | | Tool nose radius compensation right | 83, 83, 87 |
| G50* | 11 | Scaling cancel | 112 |
| G51 | | Scaling | 112 |
| G50.1* | 18 | Programable mirror image cancel | 112 |
| G51.1 | | Programable mirror image | 112 |
| G51.2 | 20 | Polygonal turning on | 179 |
| G50.2 | | Polygonal turning off | 179 |
| G52 | 00 | Local coordinate system setting | 61 |
| G53 | | Positioning in the machine coordinate system | 57 |
| G54* | 14 | Work coordinate system 1 selection | 58 |
| G55 | | Work coordinate system 2 selection | 58 |
| G56 | | Work coordinate system 3 selection | 58 |
| G57 | | Work coordinate system 4 selection | 58 |
| G58 | | Work coordinate system 5 selection | 58 |
| G59 | | Work coordinate system 6 selection | 58 |
| G61 | 15 | Exact stop mode | 46 |
| G62 | | Automatic corner override mode | 47 |
| G63 | | Override inhibit | 46 |
| G64* | | Continuous cutting | 46 |
| G65 | | Simple macro call | 187 |
| G66 | | Macro modal call (A) in every motion command | 188 |
| G66.1 | | Macro modal (B) call from each block | 189 |
| G67 | | Macro modal call (A/B) cancel | 188 |
| G68 | 16 | Mirror image for double turret on | 111 |
| G69* | | Mirror image for double turret off | 111 |
| G70 | 00 | Finishing cycle | 143 |
| G71 | | Stock removal in turning cycle | 134 |
| G72 | | Stock removal in facing cycle | 139 |
| G73 | | Pattern repeating cycle | 141 |
| G74 | | End face peck drilling cycle | 144 |
| G75 | | Outer diameter/internal diameter drilling cycle | 146 |
| G76 | | Multiple thread cutting cycle | 148 |
| G77 | 01 | Cutting cycle | 126 |
| G78 | | Thread cutting cycle | 128 |

| G code | Group | Function | Page |
|---|---|---|---|
| G79 | | End face turning cycle | 130 |
| G80* | 09 | Canned cycle for drilling cancel | 164 |
| G81 | | Drilling, spot boring cycle, | 164 |
| G82 | | Drilling, counter boring cycle | 165 |
| G83 | | Peck drilling cycle | 166 |
| G83.1 | | High Speed Peck Drilling Cycle | 160 |
| G84 | | Tapping cycle | 167 |
| G84.1 | | Counter tapping cycle | 161 |
| G84.2 | | Rigid tap cycle | 168 |
| G84.3 | | Rigid counter tap cycle | 168 |
| G85 | | Boring cycle | 171 |
| G86 | | Boring Cycle Tool Retraction with Rapid Traverse | 172 |
| G86.1 | | Fine boring cycle | 162 |
| G87 | | Boring Cycle/Back Boring Cycle | 173 |
| G88 | | Boring Cycle (Manual Operation on the Bottom Point) | 175 |
| G89 | | Boring Cycle (Dwell on the Bottom Point, Retraction with Feed) | 176 |
| G90* | 03 | Absolute command | 35 |
| G91* | | Incremental command | 35 |
| G92 | 00 | Work coordinates change/maximum spindle speed setting | 60 |
| G94* | 05 | Feed per minute | 42 |
| G95* | | Feed per revolution | 42 |
| G96 | 13 | Constant surface speed control | 65 |
| G97* | | Constant surface speed control cancel | 65 |
| G98* | 10 | Canned cycle initial level return | 155 |
| G99 | | Canned cycle R point level return | 155 |

☞ *Notes*:

– The * marked G codes in a group represent the state assumed by the control system after power-on.

– If several codes are marked with * in a group, a parameter can be set to select the effective one after power-on. They are : G00, G01; G17, G18; G43, G44, G49; G90, G91; G94, G95.

– At the time of power-on, the particular one of G20 and G21 will be effective, that has been set at the time of power-off.

– Default interpretation of command G05.1 after power-on can be specified with the *MULBUF* parameter.

– G codes in group 00 are not modal ones; the rest are so.

– More than one G code can be written in a block with the restriction that only one of the same function group may used.

– Reference to an illegal G code or specification of several G codes belonging to the same group within a particular block will produce error message *3005 ILLEGAL G CODE*.

# 4 The Interpolation

## 4.1 Positioning (G00)

The series of instructions

**G00** v

refers to a positioning in the current coordinate system.
It moves to the coordinate v. Designation v (vector) refers here (and hereinafter) to all controlled axes used on the machine-tool. (They may be X, Y, Z, U, V, W, A, B, C) E.g.:

```
G00 X(U)__ Z(W)__
```

where X, Z refer to absolute movement, while U, W refer to incremental one (provided U, W are not selected for axis).
The positioning is accomplished along a straight line involving the simultaneous movements of all axes specified in the block. The coordinates may be absolute or incremental data.
The speed of positioning cannot be commanded in the program because it is accomplished with different values for each axis, set by the builder of machine-tool as a parameter. When several axes are being moved at a time, the vectorial resultant of speed is computed by the control system in such a way that positioning is completed in a minimum interval of time, and the speed will not exceed anywhere the rapid traverse parameter set for each axis.
In executing the G00 instruction, the control system performs acceleration and declaration in starting and ending the movements, respectively.

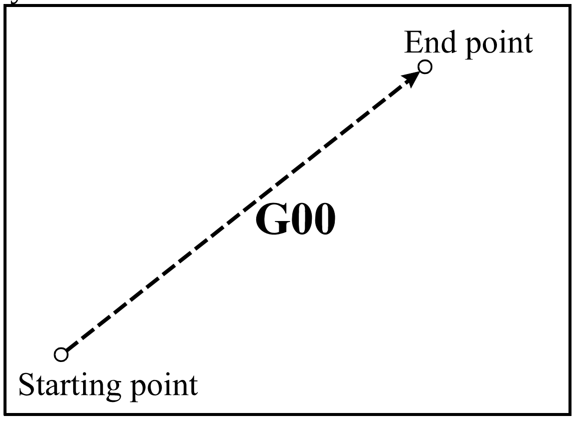


Fig. **4.1-1**

On completion of the movement, the control will check the "in position" signal when parameter *POSCHECK* in the field of parameters is 1, or will not do so when the parameter is set to 0. It will wait for the "in position" signal for 5 seconds, unless the signal arrives, the control will return the *1020 POSITION ERROR* message. The maximum acceptable deviation from the position can be specified in parameter *INPOS*.
Being a modal code, G00 remains effective until it is re-written by another interpolation command. After power-on, G00 or G01 is effective, depending on the value set in parameter group *CODES*.

## 4.2 Linear Interpolation (G01)

The series of instructions

**G01** v F

will select a linear interpolation mode. The data written for v may be absolute or incremental values, interpreted in the current coordinate system. The speed of motion (the feed) can be programmed at address F.
The feed programmed at address F will be accomplished invariably along the programmed path. Its axial components:
Feed along the axis X is

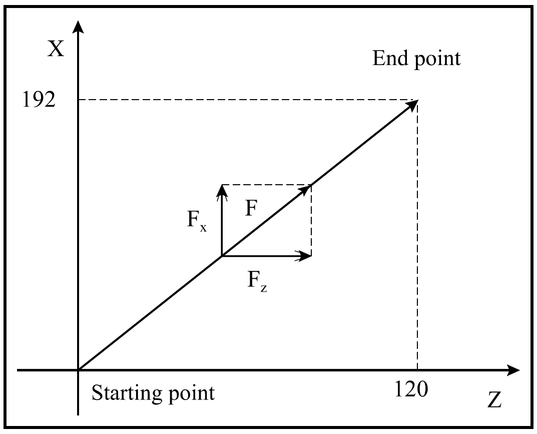


Fig. **4.2-1**

$$F_x = \frac{x}{L} F$$

Feed along the axis Z is

$$F_z = \frac{Z}{L} F$$

where x, z are the displacements programmed along the respective axes, L is the vectorial length of programmed displacement:

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

```
        G01 X192 Z120 F0.15
```

The feed along a rotational axis is interpreted in units of degrees per minute (°/min):

```
        G01 C270 F120
```

In the above block, F120 means 120deg/minute.
If the motion of a linear and a rotary axis is combined through linear interpolation, the feed components will be distributed according to the above formula. E.g. in block

```
        G91 G01 Z100 C45 F120
```

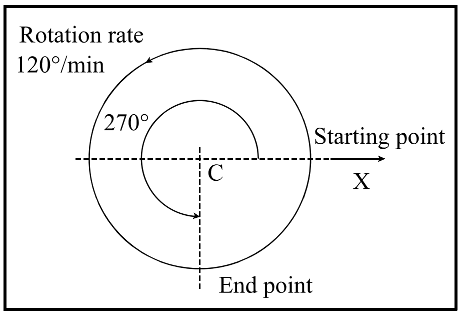


Fig. **4.2-2**

feed components in Z and B directions are:

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

Feed along axis C: $F_c = \frac{45}{\sqrt{100^2+45^2}} 120 = 49.2$ °/min

Being a modal code, G01 is effective until rewritten by another interpolation command. After power-on, G00 or G01 is effective, depending on the parameter value set in group *CODES* of the parameter field.

## 4.3 Circular and Spiral Interpolation (G02, G03)

$$G17 \begin{Bmatrix} G02 \\ \hline G03 \end{Bmatrix} X_p \; Y_p \begin{Bmatrix} R \\ \hline I \; J \end{Bmatrix} F$$

$$G18 \begin{Bmatrix} G02 \\ \hline G03 \end{Bmatrix} X_p \; Z_p \begin{Bmatrix} R \\ \hline I \; K \end{Bmatrix} F$$

$$G19 \begin{Bmatrix} G02 \\ \hline G03 \end{Bmatrix} Y_p \; Z_p \begin{Bmatrix} R \\ \hline J \; K \end{Bmatrix} F$$

The series of instructions specify circular interpolation.

A circular interpolation is accomplished in the plane selected by commands G17, G18, G19 in clockwise or counter-clockwise direction (with G02 or G03, respectively).

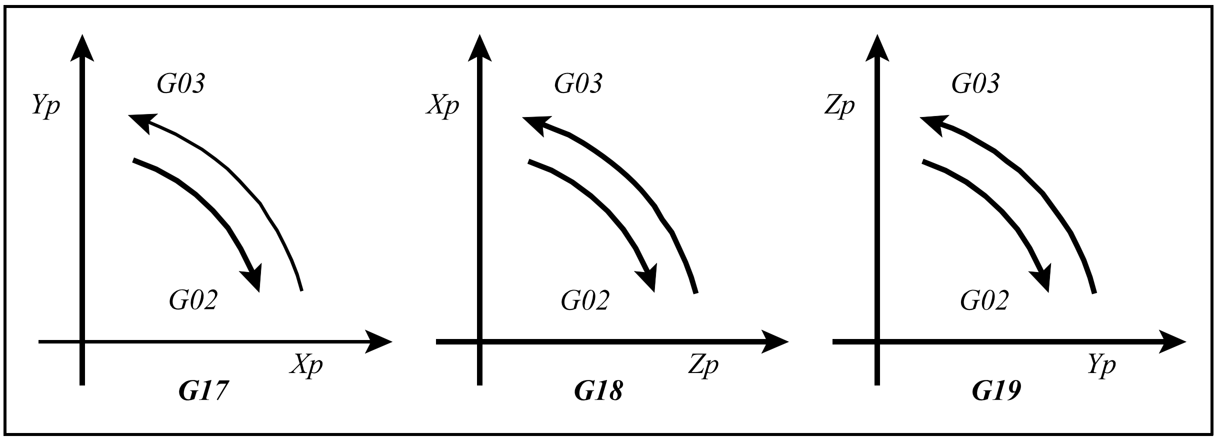


Fig. **4.3-1**

The above figure shows clockwise (G02) and counter clockwise (G03) circular directions in plane G18 when the plane is viewed in the positive-to-negative direction of axis Y. If the plane is viewed in the negative-to-positive direction of axis Y the interpretation of circular directions is reversed owing to tool turret arrangements.

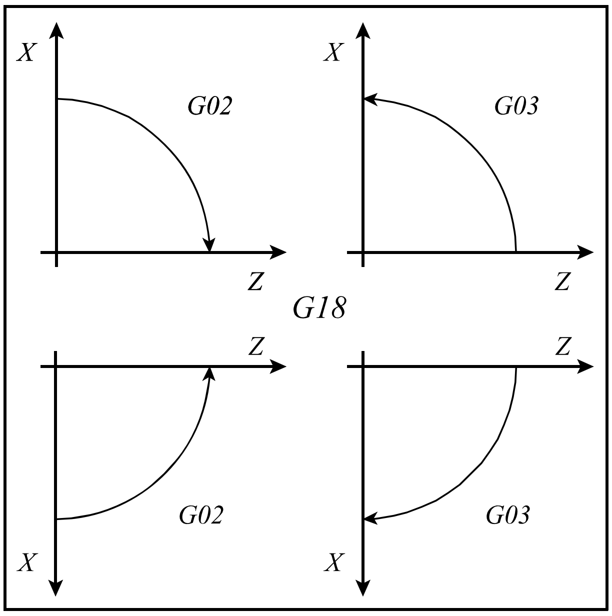


Fig. **4.3-2**

Here and hereinafter, the meanings of $X_p$, $Y_p$, and $Z_p$ are:

$X_p$: Axis X or its parallel axis,
$Y_p$: Axis Y or its parallel axis,
$Z_p$: Axis Z or its parallel axis.

The values of $X_p$, $Y_p$, and $Z_p$ are the end-point coordinates of the circle in the given coordinate system, specified as absolute or incremental data.

Further data of the circle may be specified in one of two different ways.

*Case 1*

At address R where R is the radius of the circle. Now the control will automatically calculate the coordinates of the circle center from the start point coordinates (the point where the control is in the instant of the circle block being entered), the end point coordinates (values defined at addresses $X_p$, $Y_p$, $Z_p$) and from the programmed circle radius R. Since two different circles of radius R can be drawn between the start and the end points for a given direction of circumventing (G02 or G03), the control will interpolate an arc smaller or larger than 180° when the radius of the circle is specified as a positive or a negative number, respectively. For example:

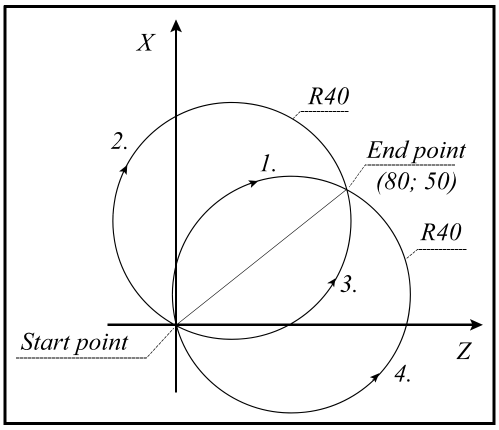


Fig. **4.3**-3

Arc section 1: `G02 X80 Z50 R40`
Arc section 2: `G02 X80 Z50 R-40`
Arc section 3: `G03 X80 Z50 R40`
Arc section 4: `G03 X80 Z50 R-40`

*Case 2*

The circle center is specified at address I, J, K for the $X_p$, $Y_p$ and $Z_p$ axes. The values specified at addresses I, J, K are interpreted always incrementally by the control system, so that the vector defined by the values of I, J, K points from the start point to the center of the circle. Value I must always be specified in radius even if X coordinate is set to diameter. For example:

With G17: `G03 X10 Y70 I-50 J-20` (X programmed in radius)
With G18: `G03 X70 Z10 I-20 K-50` (X programmed in radius)
With G19: `G03 Y10 Z70 J-50 K-20`

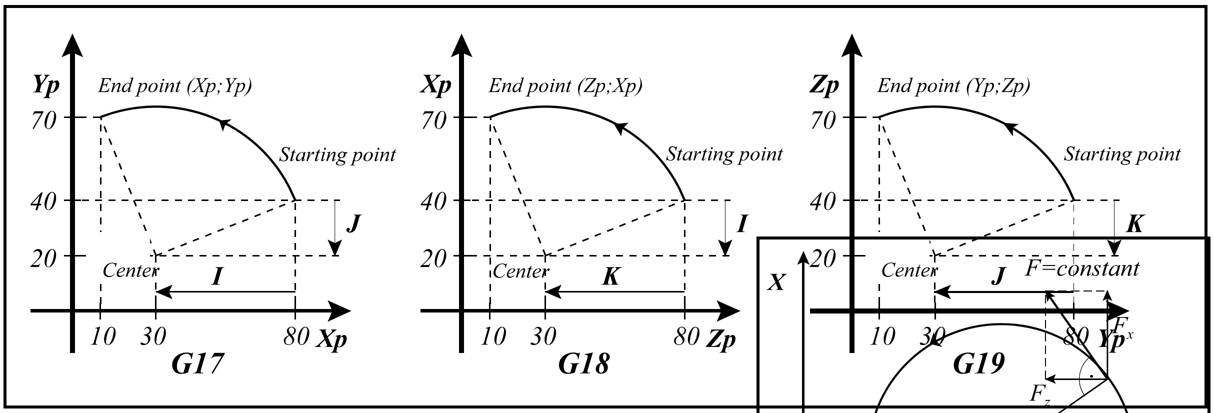


Fig. **4.3**-4

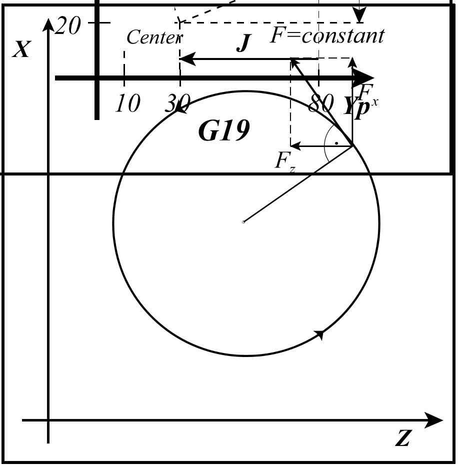


Fig. **4.3**-5

The feed along the path can be programmed at address F, pointing in the direction of the circle tangent, and being constant all along the path.

☞ *Notes*:

– I0, J0, K0 may be omitted, e.g.

```
G03 X0 Z100 I-100
```

– When each of $X_p$, $Y_p$ and $Z_p$ is omitted, or the end point coordinate coincides with the start point coordinate, then:
  a. If the coordinates of the circle center are programmed at addresses, I, J, K the control will interpolate a complete circle of 360°. E.g.:
```
G03 I-100,
```
  b. If radius R is programmed, the control returns error *3012 ERRONEOUS CIRCLE DEF. R*.
– When the circle block
  a. does not contain radius (R) or I, J, K either,
  b. or reference is made to address I, J, K outside the selected plane, the control returns *3014 ERRONEOUS CIRCLE DEF.* error. E.g. G03 X0 Y100, or (G18) G02 X0 Z100 J-100.
– The control returns error message *3011 RADIUS DIFFERENCE* whenever the difference between the start-point and end-point radii of the circle defined in block G02, G03 exceeds the value defined in parameter *RADDIF*.

Whenever the difference of radii is smaller than the value specified in the above parameter, the control will move the tool along a spiral path in which the radius is varying linearly with the central angle.
The angular velocity, not the one tangential to the path will be constant in the interpolation of a circle arc of a varying radius.

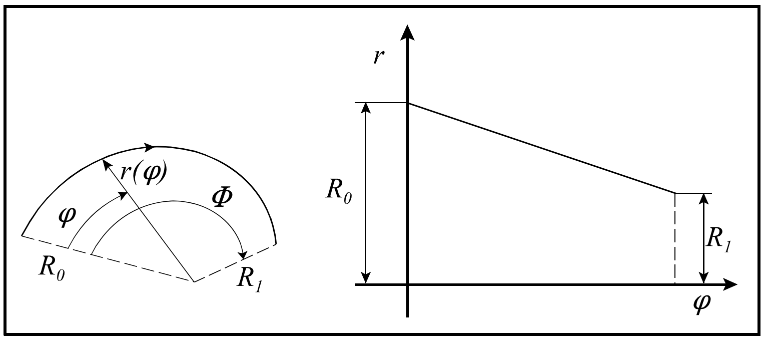


Fig. **4.3**-6

The program detail below is an example of how a spiral interpolation (circle of varying radius) can be specified by the use of addresses I, K.

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

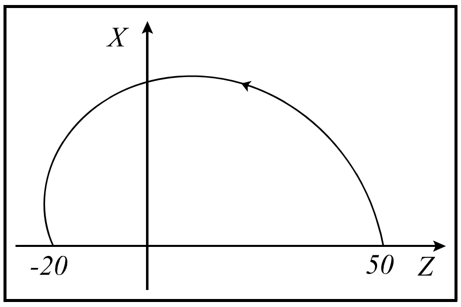


Fig. **4.3**-7

If the specified circle radius is smaller than half the distance of straight line inter-connecting the start point with the end point, the control will regard the specified radius of the circle as the start-point radius, and will interpolate a circle of a varying radius (spiral), whose center point is located on the straight line connecting the start point with the end point, at distance R from the start point.

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

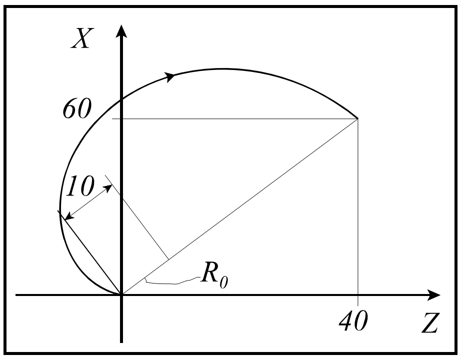


Fig. **4.3-8**

In the following sample blocks X coordinate is in diameter and U and W are assumed not to be selected for axis:

```
      G2 G90 X100 Z40 R41.2
or    G2 G90 X100 Z40 I40 J10
or    G2 G91 X60 Z30 R41.2
or    G2 (G90) U60 W30 R41.2
or    G2 (G90) XI60 ZI30 R41.2
or    G2 G91 X60 Z30 I40 J10
or    G2 (G90) U60 W30 I40 J10
or    G2 (G90) XI60 ZI30 I40 J10
```

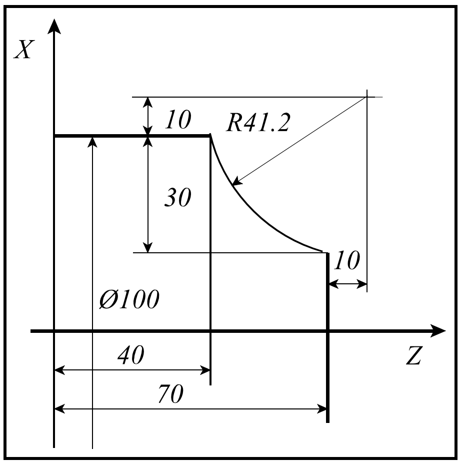


Fig. **4.3-9**

## 4.4 Equal Lead Thread Cutting (G33)

The instruction

**G33** v F Q
**G33** v E Q

will define a straight or taper thread cutting of equal lead.

The coordinates of maximum two axes can be written for vector v. The control will cut a tapered thread if two coordinated data are assigned to vector v. The control will take the lead into consideration along the long axis.

If α<45°, i.e. Z>X, the programmed lead will be taken into account along axis Z,

if α>45°, i.e. X>Z, the control will take the programmed lead along axis X.

The lead can be defined in one of two 2 ways.

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

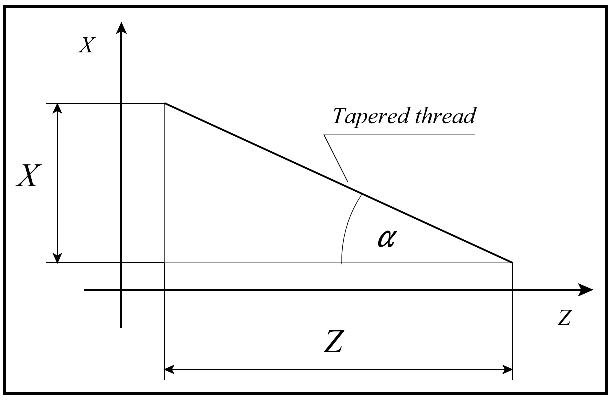


Fig. **4.4**-1

– If the pitch is specified at address E, the control will cut an inch thread. Address E is interpreted as number of ridges per inch. If, e.g., E3 is programmed, the control will cut a thread ⅓"=25.4/3=8.4667mm lead.

The shift angle of the thread start is specified at address Q expressed in degrees from the zero pulse of the spindle encoder. A multiple 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 the various threads are to be cut. If, e.g., a double thread is to be cut, the first and the second starts will be commenced from Q0 (no special programming is needed) and from Q180, respectively.

G33 is a modal function. If several thread-cutting blocks are programmed in succession, threads can be cut in any arbitrary surface limited by straight lines.

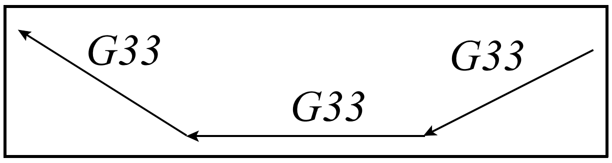


Fig. **4.4**-2

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 resulting in a continuous thread in each section of lines. Hence the programmed shift angle of the thread start (Q) will also be taken into account in the first block.

An example of programming a thread-cutting:

```
G0 G90 X50 Z40
U-30
G33 U10 W38 F2
G0 U20
W-38
```

In the example above X is specified in diameter.

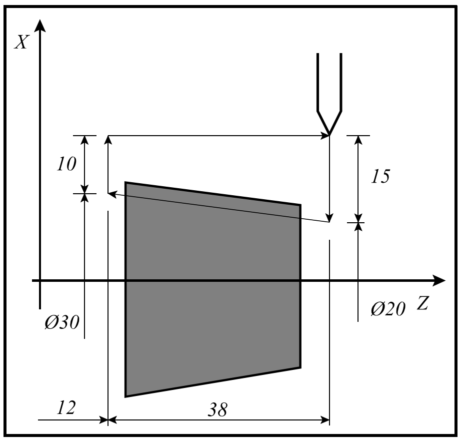


Fig. **4.4**-3

☞ *Notes*:

– The control returns error message *3020 DATA DEFINITION ERROR G33* if more than two coordinates are specified at a time in the thread-cutting block, or if both addresses F and E are specified simultaneously.

– Error message *3022 DIVIDE BY 0 IN G33* is produced when 0 is specified for address E in the thread-cutting block.

– An encoder has to be mounted on the spindle for the execution of command G33.

– In the course of command G33 being executed, the control will take the feed and spindle override values automatically to be 100%; the effect of the stop key will only prevail after the block has been executed.

– On account of the following error of the servo system, overrun and run out allowances have to be provided for the tool in addition to the part at the beginning and end of the thread in order to obtain a constant lead all along the part.

– In the course of thread-cutting the feed (in mm/minute) may not exceed the value selected in the group of parameters FEEDMAX.

– In the course of thread-cutting the speed (r.p.m) of the spindle may not exceed the maximum speed permissible for the spindle encoder mechanically and electrically (the maximum output frequency).

## 4.5 Variable-Lead Thread Cutting (G34)

Command

**G34** v F Q K

cuts straight or tapered threads of variable-lead. The interpretation of input data v, F, Q corresponds to those written for function G33. The interpretation of K:

**K**: Increase or decrease of thread lead per spindle revolution.

The value of K ranges from 0.001 mm/rev (0.0001 inch/rev) to 500 mm/rev (10 inch/rev).

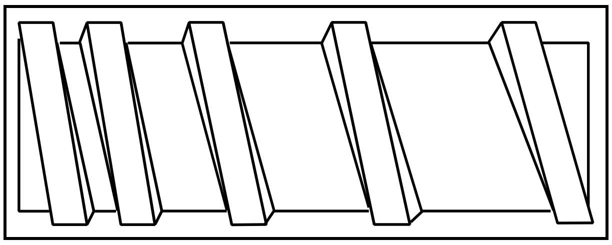


Fig. **4.5-1**

## 4.6 Polar Coordinate Interpolation (G12.1, G13.1)

Polar coordinate interpolation is a control operation method, in case of which the work described in a Cartesian coordinate system moves its contour path by moving a linear and a rotary axis.
Command

**G12.1**

switches polar coordinate interpolation mode on. The path of the milling tool can be described in the succeeding part program in a Cartesian coordinate system in the usual way by programming linear and circular interpolation, by taking the tool radius compensation into account. *The command must be issued in a separate block and no other command can be written beside.*
Command

**G13.1**

switches polar coordinate interpolation mode off. *The command must be issued in a separate block and no other command can be written beside.* It always registers state G13.1 after power-on or reset.

Plane selection

A plane determining the address of the linear and the rotary axis to be applied must be selected before switching polar coordinate interpolation on.

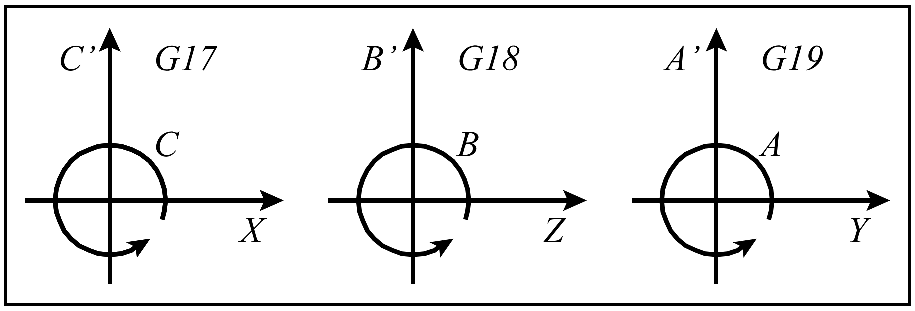


Fig. **4.6-1**

Command

G17 X_ C_

selects axis X for linear axis, while as for the rotary axis it is axis C. The virtual axis is indicated with C' on the diagram, the programming of which is implemented by defining length measures. With the help of commands

G18 Z_ B_
G19 Y_ A_

further linear and rotary axes can be selected together in the above mentioned way.

Work zero point offset in the course of polar coordinate interpolation

In case of using polar coordinate interpolation the origin of the applied work coordinate system on axis X must be chosen so that the rotation axis of the turning tool in position X=0 coincides with the rotation axis of the circular axis (C).

Using tool length compensation during polar coordinate interpolation

***The compensation value of the tool used in polar coordinate interpolation must be 0 on axis X***, supposingly that axes X and C take part in the interpolation. Compensation on axis X must be set by work zero point offset (see the item above).

Position of the axes when polar coordinate interpolation is switched on

Before switching polar coordinate interpolation on (command G12.1) make sure that the ***circular axis position is 0***. The ***linear axis position*** can either be negative or positive but it ***cannot be 0***.

Programming length coordinates 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. If e.g. axes X and C have been selected by means of command G17 X_ C_ address C can be programmed like axis Y in the case of plane selection G17 X_ Y_. The programming of the first axis being in diameter does not influence the programming of the ***virtual axis***, the coordinate data must always be given in ***radius*** for the ***virtual axis***. If, e.g., polar coordinate interpolation is executed in plane X C the value written at address C must be specified in radius, independent of address X given in diameter or radius.

Move of axes not taking part in polar coordinate interpolation

The tool on these axes moves normally, independent of the switched-on state of the polar coordinate interpolation.

Programming circular interpolation in the course of polar coordinate interpolation

Definition of a circle in polar coordinate interpolation mode is possible as known by means of the radius or by programming the circle center coordinates. In the latter case addresses I, J, K must be used according to the selected plane as seen below:

| | | |
|---|---|---|
| G17 X_ C_ | G18 Z_ B_ | G19 Y_ A_ |
| G12.1 | G12.1 | G12.1 |
| ... | ... | ... |
| G2 (G3) X_ C_ I_ J_ | G2 (G3) B_ Z_ I_ K_ | G2 (G3) Y_ A_ J_ K_ |

Use of tool radius compensation in case of polar coordinate interpolation

Commands G41, G42 can be used customary in polar coordinate interpolation. Beware of the value of imaginary tool nose number in the compensation group of the rotating tool must be Q=0. The following restrictions must be considered regarding its application:

– Switch-on of polar coordinate interpolation (command G12.1) is only possible in state G40,
– If G41 or G42 is switched on in state G12.1, G40 must be programmed before switching polar coordinate interpolation off (command G13.1).

Programming restrictions in the course of polar coordinate interpolation

The below commands cannot be used in the switched-on state of polar coordinate interpolation:

– plane change: G17, G18, G19,
– coordinate transformations: G52, G92,
– work coordinate system change: G54, ..., G59,
– orientation in machine coordinate system: G53.

Feed in the course of polar coordinate interpolation

Interpretation of feed in polar coordinate interpolation is tangential speed as in case of right angle interpolation: The relative speed of piece and tool is defined.

With polar coordinate interpolation the path described in a Cartesian coordinate system is done by moving a linear and a rotary axis. As the tool center approaches the circular axis of rotation, the rotary axis should have to take larger and larger steps within a time unit so that the path speed is constant. However the maximum speed permitted for the rotary axis defined by parameter limits circular axis speed. Therefore, near to the origin the control decreases feed step by step for the rotary axis speed not to exceed all limits. The diagram beside shows the cases when straight lines parallel to axis X (1, 2, 3, 4) are programmed. Δx move belongs to the programmed feed within a time unit. Different angular moves ($\varphi_1$, $\varphi_2$, $\varphi_3$, $\varphi_4$) belong to Δx move for each straight lines (1, 2, 3, 4). Apparently, the closer the machining gets to the origin the larger angular movement the rotary axis has to make within a time unit in order to keep the programmed feed.

In case the angular move to be made within a time unit exceeds the value of parameter FEEDMAX set for rotary axis the control gradually decreases the tangential feed.

With these in mind, programs in case of which the tool center moves close to the origin are to be avoided.

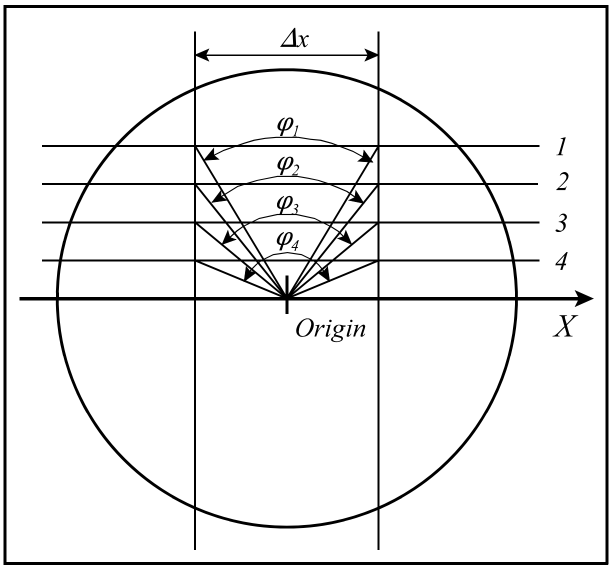


Fig. **4.6-2**

Example

Below an example for the use of polar coordinate interpolation is shown. The axes taking part in the interpolation: Axes X (linear axis) and C (rotary axis). Axis X is programmed in diameter, while that of axis C is in radius.

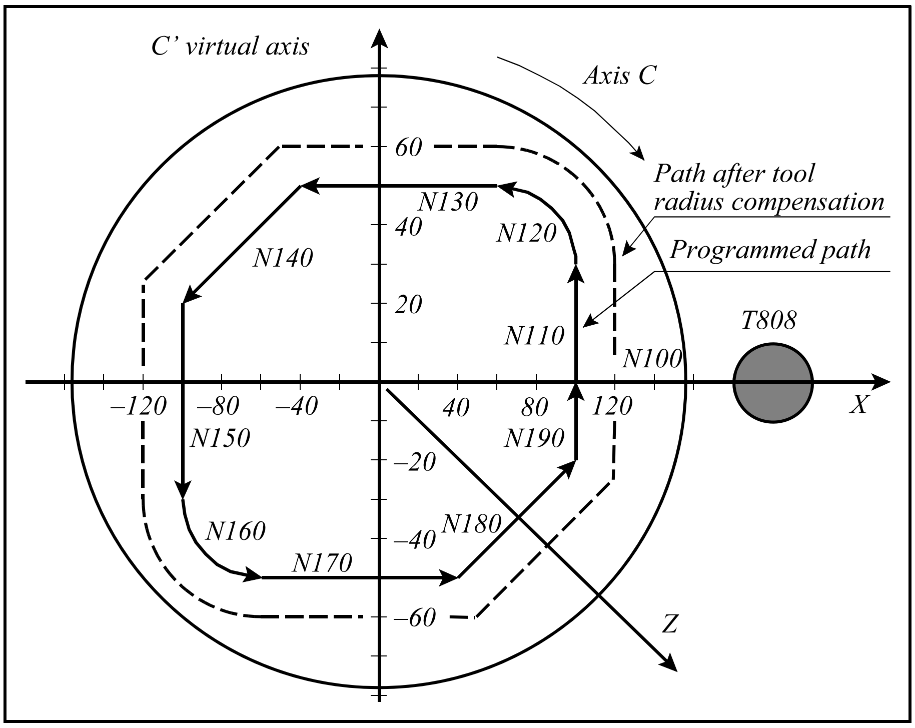


Fig. **4.6-3**

```
%O7500(POLAR COORDINATE INTERPOLATION)

...
N050 T808
N060 G59                      (start position of coordinate system G59 in
                              direction X on rotary axis C)
N070 G17 G0 X200 C0           (select plane X, C; orientation to 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              (Retract tool, select plane X, Z)
...

%
```

## 4.7 Cylindrical Interpolation (G7.1)

Should a cylindrical cam grooving be milled on a cylinder mantle, cylindrical interpolation is to be used. In this case the rotation axis of the cylinder and of a rotary axis must coincide. The rotary axis movements are specified in the program in degrees, which are converted into linear movement along the mantle by the control in function of the cylinder radius, so that linear and circular interpolation can be programmed together with another linear axis. The movements resulted after the interpolations, are re-converted into movement in degrees for the rotary axis.
Command cylindrical interpolation on

**G7.1** Qr

switches cylindrical interpolation on, where

**Q**: address of rotary axis taking part in the cylindrical interpolation,
**r**: cylinder radius.

If for example the rotary axis acting in cylindrical interpolation is axis C and the cylinder radius is 50 mm, cylindrical interpolation is switched on by means of command G7.1 C50.
In the succeeding part 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 be given in mm, while that of the rotary axis in degrees (°).
Command cylindrical interpolation off

**G7.1** Q0

switches cylindrical interpolation off, i.e. code G corresponds to that of the switch-on, except for the address of rotary axis being 0.
The cylindrical interpolation indicated in the above example (G7.1 C50) can be switched off with the help of command G7.1 C0.
Command G7.1 must be issued in a separate block.

Plane selection

The plane selection code is always determined by the name of the linear axis parallel to the rotary axis. The rotary axes parallel to axes X, Y and Z are axes A, B and C, respectively.

| | | |
|---|---|---|
| G17 X A or | G18 Z C or | G19 Y B or |
| G17 B Y | G18 A X | G19 C Z |

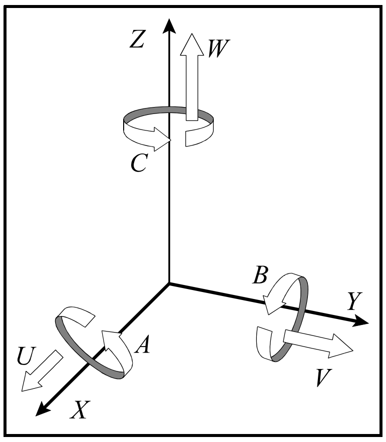


Fig. 4.7-1

Circular interpolation

It is possible to define circular interpolation in cylindrical interpolation mode, however only by specifying radius R.
*No circular interpolation can be executed in case of cylindrical interpolation by giving the circle center (I, J, K).*
The circle radius is always interpreted in mm or 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 (G3) Z_ C_ R_ | G2 (G3) C_ Z_ R_ |

Application of tool radius compensation in case of cylindrical interpolation

Commands G41, G42 can be used in the usual manner in the switched-on state of cylindrical interpolation. Though the following restrictions are in effect regarding its application:
– Switch-on of cylindrical interpolation (command G7.1 Qr) is only possible in state G40.

– Should G41 or G42 be switched on in cylindrical interpolation mode, G40 must be programmed before switching cylindrical interpolation off (command G7.1 Q0).

Programming restrictions in the course of cylindrical interpolation

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

Example

The diagram beside shows a path milled 3 mm deep on the mantle of an R=28.65-mm-radial cylinder. Rotating tool T606 is parallel to the axis X.. 1° movement on the cylinder mantle is:

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

The axis order seen on the diagram corresponds to plane selection G19.

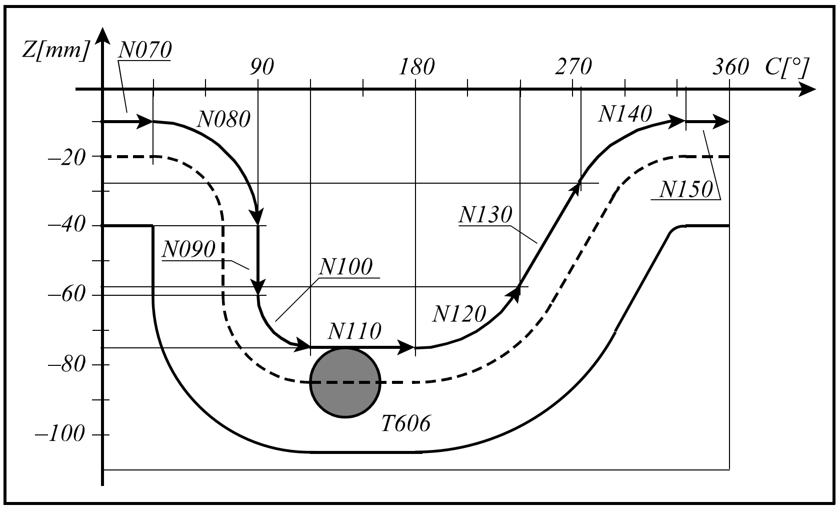


Fig. 4.7-2

```
%O7602(CYLINDRICAL INTERPOLATION)
...
N020 G0 X200 Z20 S500 M3 T606
N030 G19 Z-20 C0                    (G19: select plane C–Z)
N040 G1 X51.3 F100
N050 G7.1 C28.65                    (cylindrical  interpolation on, rotary
                                    axis: C, cylinder radius: 28.65mm)
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                        (cylindrical interpolation off)
N180 G0 X100
...
%
```

# 5 The Coordinate Data

## 5.1 Absolute and Incremental Programming (G90, G91), Operator I

The input coordinate data can be specified as absolute or incremental values. In an absolute specification, the coordinates of the end point have to be specified for the control, for an incremental data, it is the distance to go in the block.

**G90**: Programming of absolute data

**G91**: Programming of incremental data

G90 and G91 are modal functions. Parameter group *CODES* will decide which state will be assumed by the control system at the time of power-on.

Movement to an absolute position is only feasible after a reference point return.

Example:

As shown in the Figure, a movement can be programmed in one of two different ways.

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

Operator I will be effective under the conditions of an absolute data specification (G90). It is only applicable to the coordinate, whose address precedes it. It means an incremental data. The alternative way of solving the above example:

```
(G90) G01 XI60 ZI–40
G01 XI60 Z20
G01 X100 ZI–40
```

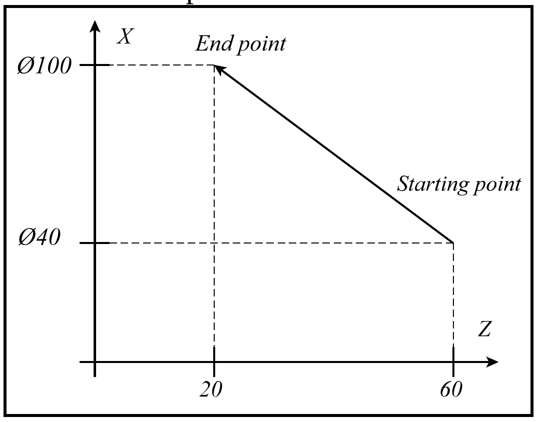


Fig. **5.1-1**

Unless addresses U, V, W are selected for an axis they can be used for indicating incremental movement in X, Y, Z directions, respectively.

| | Address of absolute command | Address of incremental command |
|---|---|---|
| Motion command in X direction | X | U |
| Motion command in Z direction | Z | W |
| Motion command in C direction | C | H |
| Motion command in Y direction | Y | V |

The sample example with the above taken into account:

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

## 5.2 Inch/Metric Conversion (G20, G21)

With the appropriate G code programmed, the input data can be specified in metric or inch units.

**G20**: Inch input programming

**G21**: Metric input programming

At the beginning of the program, the desired input unit has to be selected by specifying the appropriate code. The selected unit will be effective until a command of opposite meaning is issued, i.e., G20 and G21 are modal codes. Their effect will be preserved even after power-off, i.e., the

unit prevailing at the time of power-off will be effective after power-on.
*The change of the unit will affect the following items:*

- *Coordinate and compensation data,*
- *Feed,*
- *Constant surface speed*
- *Position, compensation and feed displays.*

## 5.3 Specification and Value Range of Coordinate Data

Coordinate data can be specified in 8 decimal digits.
The decimal point will be interpreted as the function of the unit of measure applied:
– X2.134 means 2.134 mm or 2.134 inch,
– B24.36 means 24.36 degrees when address B refers to a rotary axis.
The use of a decimal point is not mandatory.
– X325 means e.g. 325 mm.
The leading zeros may be omitted.
– .032=0.032
The trailing zeros may be omitted behind the decimal point.
– 0.320=.32
The control will interpret a number with more decimals defined by the increment system. For example, command X1.23456 will be, when IS-B increment system is selected, interpreted as
– 1.235 mm (metric unit),
– 1.2346 inch (inch unit).
Accordingly, the input data will be output as rounded values.
The value ranges of the length coordinates are shown in the Table below.

| input unit | output unit | increment system | value range of length coordinates | unit of measure |
|---|---|---|---|---|
| mm | mm | IS-A | ± 0.01-999999.99 | mm |
| | | IS-B | ± 0.001-99999.999 | |
| | | IS-C | ± 0.0001-9999.9999 | |
| inch | mm | IS-A | ± 0.001-39370.078 | inch |
| | | IS-B | ± 0.0001-3937.0078 | |
| | | IS-C | ± 0.00001-393.70078 | |
| inch | inch | IS-A | ± 0.001-99999.999 | inch |
| | | IS-B | ± 0.0001-9999.9999 | |
| | | IS-C | ± 0.00001-999.99999 | |
| mm | inch | IS-A | ± 0.01-999999.99 | mm |
| | | IS-B | ± 0.001-99999.999 | |
| | | IS-C | ± 0.0001-9999.9999 | |

The value ranges of angular coordinates:

| increment system | value range of angular coordinates | unit of measure |
|---|---|---|
| IR-A | ± 0.01-999999.99 | degrees |
| IR-B | ± 0.001-99999.999 | |
| IR-C | ± 0.0001-9999.9999 | |

## 5.4 Programming in Radius or Diameter

Since the section of a turned work is generally a circle, sizes in direction of X axis can also be specified in diameter. The following parameter determines if the X-direction size is interpreted by the control in radius or diameter:
In case of **programming in radius**:
4762 DIAM=0
In case of **programming in diameter**:
4762 DIAM=1
When programing in diameter the least input increment programable is 1. In this case the control steps ½ increments. E.g., if INCRSYSTB=1 the least command increment is 0.001 mm and the control steps 0.0005 mm in radius.
If the parameter is set to programming diameters, the below cases must be taken into account:

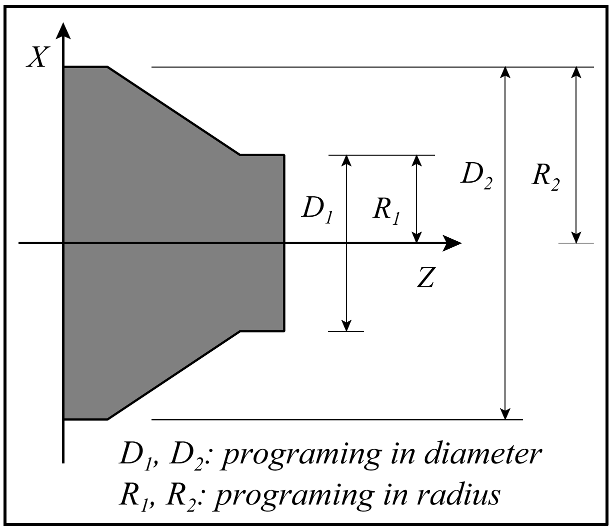

X
Z
$D_1$
$R_1$
$D_2$
$R_2$
$D_1$, $D_2$: programing in diameter
$R_1$, $R_2$: programing in radius

Fig. 5.4-1

| Case | Note |
|---|---|
| Absolute move command in X direction | Specified with a **diameter** value |
| Incremental move command in X direction | Specified with a **diameter** value ($D_2$ –$D_1$ in the figure) |
| X component of coordinate system offset | Specified with a **diameter** value |
| X component of tool length offset | Specified with a **diameter** value |
| Parameters in canned cycles, such as cutting depth along X axis | Always specified with a **radius** value |
| Radius designation in circular interpolation (R, I) | Always specified with a **radius** value |
| Display of X axis position | Displayed as **diameter** value |
| Feedrate along X axis in facing | Always specified in **radius/rev** or **radius/min** |
| Increment size in incremental jog and handwheel mode | 1 increment=1µm in **diameter** |

## 5.5 Rotary Axis Roll-over

This function can be used in case of rotary axes, i.e., if address A, B or C is selected for operating rotary axis. Handling of roll-over means, that the position on the given axis is not registered between plus and minus infinity, but regarding the periodicity of the axis, e.g., between 0˚ and 360˚.

Selecting rotary axis

The selection can be executed by setting parameter 0182 A.ROTARY, 0185 B.ROTARY or 0188 C.ROTARY to 1 for axes A, B or C, respectively. If among these parameters one is set to 1
– the control does not execute inch/metric conversion for the appropriate axis,
– roll-over function can be enabled for that axis by setting the appropriate parameter ROLL-OVEN to 1.

Enabling the handling of roll-over

The function is affected by setting parameter 0241 ROLLOVEN_A, 0242 ROLLOVEN_B or 0243 ROLLOVEN_C to 1 for axes A, B or C, respectively, provided the appropriate axis is a rotary one. If the given parameter ROLLOVEN_x
– =0: the rotary axis is regarded as linear axis and the setting of further parameters is uneffective,
– =1: handling of roll-over is applied for the rotary axis, the essence of which is discussed below.

Specifying path per roll-over

The path per one roll-over of the axis is defined at parameter 0261 ROLLAMNT_A, 0262 ROLLAMNT_B or 0263 ROLLAMNT_C in input increment for axes A, B or C, respectively. Thus if the control is operating in increment system B and the axis rotates 360° per one roll-over, the value to be written at the appropriate parameter is 360000.
With the help of the above parameter settings the control always displays the position of the rotary axis in range 0°- +359.999° independent of the direction of rotation and the number of revolutions.

Movement of rotary axis in case of absolute programming

In case of absolute data input, when handling of roll-over is enabled for rotary axis (ROLLOVEN_x=1), the axis never moves more than that set at appropriate parameter ROLLAMNT_x. That is, if, e.g., ROLLAMNT_C=360000 (360˚), the maximum movement is 359.999°.

For the movement direction to always be according to the sign of position given at the axis address or in the shorter way can be set on the basis of parameter 0244 ABSHORT _A, 0245 ABSHORT_B or 0246 ABSHORT_C. If appropriate parameter ABSHORT_x

– =0: it always moves in the direction of the sign of the programmed position

– =1: it always moves in the shorter direction.

| 0188 **C.ROTARY**=1, 0243 **ROLLOVEN_C**=1 0263 **ROLLAMNT_C** = =360000 | Block programmed by absolute coordinate input | Movement affected by block | Position at block end |
|---|---|---|---|
| 0246 **ABSHORT_C**=0<br><br>it always moves in direction of sign programmed at address C | | | C=0 |
| | G90 C450 | 90 | C=90 |
| | G90 C0 (**0 is a positive number!**) | 270 | C=0 |
| | G90 C–90 | –90 | C=270 |
| | G90 C–360 | –270 | C=0 |
| 0246 **ABSHORT_C**=1<br><br>it always moves in the shorter direction | | | C=0 |
| | G90 C450 | 90 | C=90 |
| | G90 C0 | –90 | C=0 |
| | G90 C–90 | –90 | C=270 |
| | G90 C–360 | 90 | C=0 |

Movement of rotary axis in case of incremental programming

In case of programming incremental data input the direction of movement is always according to the programmed sign.

The appropriate parameter ROLLAMNT_x to be applied for movement setting can be set at parameter 0247 RELROUND_A, 0248 RELROUND_B or 0249 RELROUND_C for axis A, B or C, respectively. If the appropriate parameter RELROUND_x

– =0: parameter ROLLAMNT_x is out of use, i.e. the movement can be greater than 360˚,

– =1: parameter ROLLAMNT_x is in use. If, e.g., ROLLAMNT_C=360000 (360˚), the largest movement on axis C may be 359.999°.

| 0188 **C.ROTARY**=1,<br>0243 **ROLLOVEN_C**=1<br>0263 **ROLLAMNT_C** = =360000 | Block programmed by incremental data input | Movement affected by block | Position at block end |
|---|---|---|---|
| 0249 **RELROUND_C**=0<br><br>parameter ROLLAMNT_C is out of use | | | C=0 |
| | G91 C450 | 450 | C=90 |
| | G91 C0 | 0 | C=90 |
| | G91 C–90 | –90 | C=0 |
| | G91 C–360 | –360 | C=0 |
| 0249 **RELROUND_C**=1<br><br>parameter ROLLAMNT_C is in use | | | C=0 |
| | G91 C450 | 90 | C=90 |
| | G91 C0 | 0 | C=90 |
| | G91 C–90 | –90 | C=0 |
| | G91 C–360 | 0 | C=0 |

# 6 The Feed

## 6.1 Feed in Rapid Traverse

G00 commands a positioning in rapid traverse.
The value of rapid traverse for each axis is set by parameter by the builder of the machine. The rapid traverse may be different for each axis.
When several axes are performing rapid traverse motions simultaneously, the resultant feed will be calculated in such a way that the speed component of each axis will not exceed the particular rapid traverse value (set as a parameter), and the positioning is accomplished in a minimum of time.
Rapid traverse rate is modified by the rapid traverse override switch that can be

F0: Defined by parameter RAPOVER in %,
and 25%, 50%, 100%.

The rapid traverse rate will not exceed 100%.
Rapid traverse will be stopped if the state of the feedrate override switch is 0%.
In lack of a valid reference point, the reduced rapid traverses defined by the machine tool builder by parameter will be effective for each axis until the reference point is returned.
Rapid traverse override values can be connected to the feedrate override switch.
When the slide is being moved by the jog keys, the speed of rapid traverse is different from the rapid traverse in G00, it is also selected by parameters separately for each axis. Appropriately it is lower than the speed of positioning for human response times.

## 6.2 Cutting Feed Rate

The feed is programmed at address **F**.
The programmed feed is accomplished in blocks of linear (G01) and circular interpolations (G02, G03).
The feed is accomplished tangentially along the programmed path.

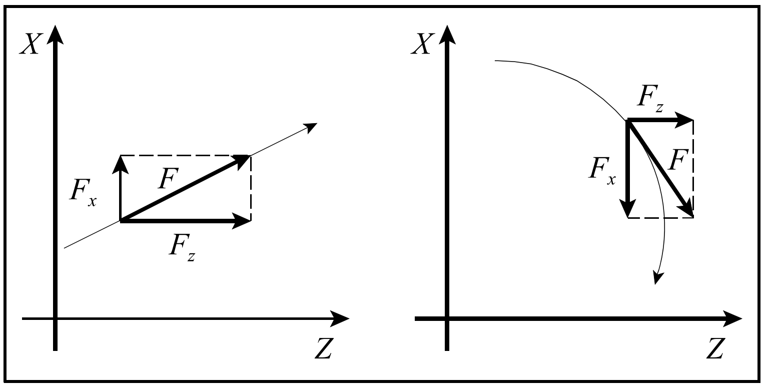


Fig. **6.2-1**

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

$$F=\sqrt{F_x^2+F_z^2}$$

Except for override and stop inhibit states (G63), the programmed feed can be modified over the range of 0 to 120% with the feed-override switch.
The feed value (F) is modal. After power-on, the feed value set at parameter *FEED* will be effective.

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

The unit of feed can be specified in the program with the G94 and G95 codes:

**G94**: Feed per minute
**G95**: Feed per revolution

The term "feed/minute" refers to a feed specified in units mm/minute, inch/minute or degree / minute.
The term "feed/rev" refers to the feed accomplished in a revolution of the spindle, in units of mm / rev, inch/minute or deg/rev. A G95 cannot be programmed unless the spindle is equipped with an encoder.
Modal values. After power-on, state G94 or G95 will be selected with reference to parameter group *CODES*. State G94/G95 will be unaffected the rapid traverse, it is invariably in units of minutes.

The Table below shows the maximum programmable range of values at address F, for various cases.

| input units | output units | increment system | value range of address F | unit |
|---|---|---|---|---|
| mm | mm | IS-A | 0.001 - 250000 | mm or deg/min |
| | | IS-B | 0.0001 - 25000 | |
| | | IS-C | 0.00001 - 2500 | |
| | | IS-A | 0.0001 - 5000 | mm or deg/rev |
| | | IS-B | 0.00001 - 500 | |
| | | IS-C | 0.000001 - 50 | |
| inch | mm | IS-A | 0.0001 - 9842.5197 | inch or deg/min |
| | | IS-B | 0.00001 - 984.25197 | |
| | | IS-C | 0.000001 - 98.25197 | |
| | | IS-A | 0.00001 - 196.85039 | inch or deg/rev |
| | | IS-B | 0.000001 - 19.685039 | |
| | | IS-C | 0.0000001 - 1.9685039 | |
| inch | inch | IS-A | 0.0001 - 25000 | inch or deg/min |
| | | IS-B | 0.00001 - 2500 | |
| | | IS-C | 0.000001 - 250 | |
| | | IS-A | 0.00001 - 500 | inch or deg/rev |
| | | IS-B | 0.000001 - 50 | |
| | | IS-C | 0.0000001 - 5 | |
| mm | inch | IS-A | 0.001 - 250000 | mm or deg/min |
| | | IS-B | 0.0001-25000 | |
| | | IS-C | 0.00001-2500 | |
| | | IS-A | 0.0001 - 5000 | mm or deg/rev |
| | | IS-B | 0.00001-500 | |
| | | IS-C | 0.000001-50 | |

### 6.2.2 Clamping the Cutting Feed

The maximum programmable feed on a particular machine can be clamped (set as a parameter) by the manufacturer of the machine. The value set there invariably refers to minutes. That value is also the speed of DRY RUN. If a programmed feed higher than that is set, the control will clamp it automatically in the course of program execution.
The maximum jog feed can also be clamped separately by parameters for human response times.

## 6.3 Acceleration/Deceleration. Taking F Feed into Account

Acceleration and deceleration in case of movement start and stop is needed in order to minimize or level the effect of powers mechanically taxing the machine.
Normally the control accelerates or decelerates in the following cases:
– in case of manual movement,
– in the course of rapid override orientation (G0), movement at the beginning of block always starts from 0, while at the end of orientation it always decelerates to 0,
– in case of feed (G1, G2, G3) in state G9 or G61 movement at the beginning of block always starts from 0, while at the end of movement it always decelerates to 0,
– in case of feed (G1, G2, G3) and more simultaneous feed blocks it accelerates at the beginning of the series of blocks and decelerates at the end,
– in the above case it also accelerates as well as decelerates between the feed blocks if it detects corner,
– in the above case it also accelerates or decelerates, if feed (F) is changed in any of the blocks or a function limiting feed is in effect in the block,
– it decelerates, if feed is stopped by means of STOP button, and accelerates, if feed is started by means of START button,
– it stops after deceleration, if function is executed after the movement as well as at the end of block if SINGLE BLOCK switch is in effect.

Not the feed components on axes but always the value of mutual (vectorial) feed is accelerated by the control. Two kinds of acceleration can be set:
– linear and
– bell-shaped curve

In case of linear acceleration the value of acceleration is constant during acceleration or deceleration, the control increases and decreases feed according to linear function in case of start and stop, respectively. Different acceleration values can be set for each axis at parameter *ACCn* in mm/sec$^2$, as requested. If more axes are involved in the movement, acceleration as well as deceleration are always executed on the basis of the parameter of the axis the acceleration of which is set to the lowest.

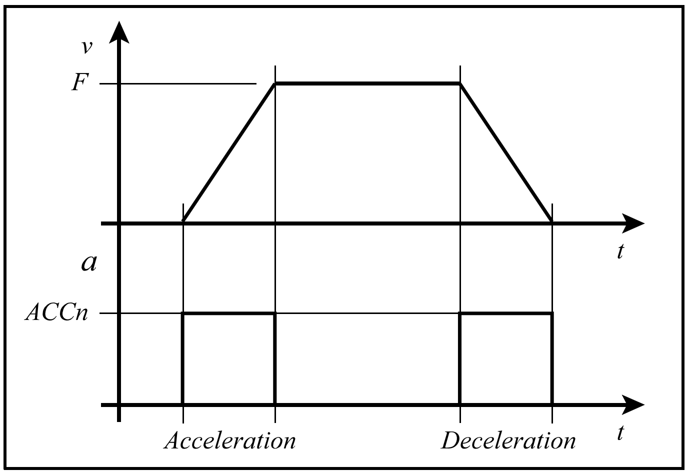


Fig. **6.3-1**

In case of bell-shaped acceleration the value of acceleration changes, i.e. it increases in the course of acceleration until it reaches the acceleration value set (*parameter ACCn*) as well as it decreases linearly before reaching the target speed. Conclusively the shape of feed rise and fall is a bell-shaped curve in function of time (quadratic curve), hence the name bell-shaped curve acceleration. T time under which the control reaches the acceleration value set can be defined differently for each axis at parameter *ACCTCn* in msec, as requested. If more axes are involved in the movement, acceleration as well as deceleration are executed on the basis of the parameter of the axis set to the highest time constant.

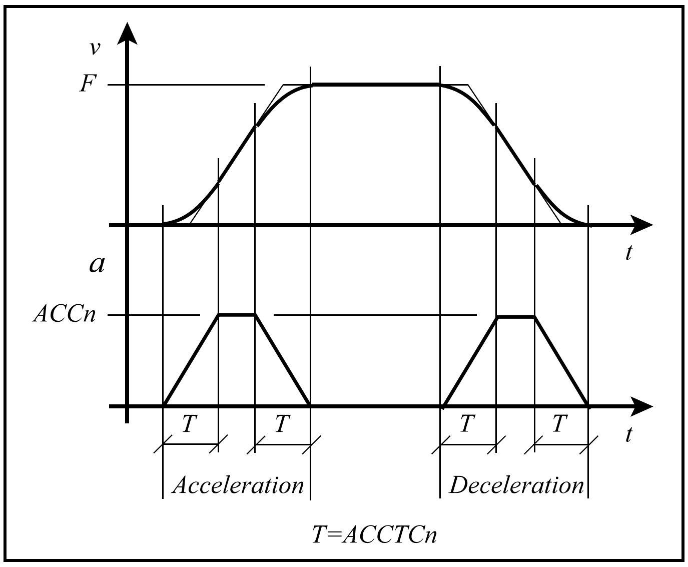


Fig. **6.3-2**

*Accelerations and time constants for each axis are always defined by the constructor of the machine in function of the dynamic load capacity of the machine.*

New acceleration to a higher-than-before feed value is executed by the control always in course of execution of the block in which the new feed has been given. This process may take over more blocks, if necessary. New deceleration to a lower-than-before feed value is started by the control in a previous suitable block, so that in the block, where the new feed has been set, machining is started at the programmed speed.

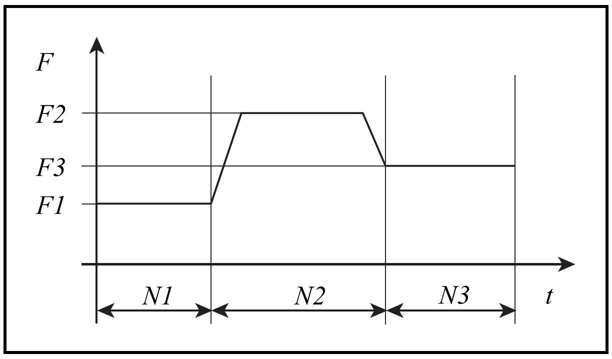


Fig. **6.3-3**

The control pre-monitors and records tangential speed changes. This is necessary to reach the target speed even with a continuous acceleration lasting over the execution of more blocks. This function is only available in MULTIBUFFER mode (value of parameter *MULBUF* is 1).

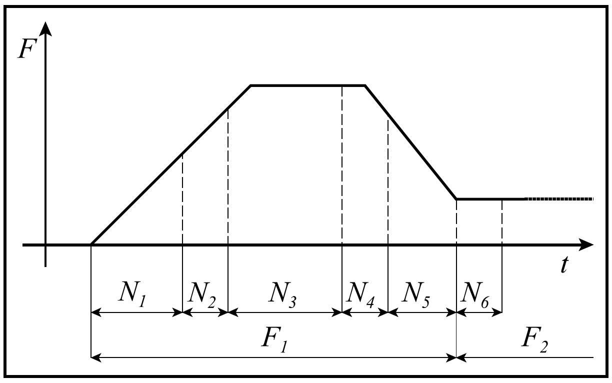


Fig. **6.3-4**

## 6.4 Feed Control Functions

The override control functions are required when corners are to be machined, and/or when the particular technology requires the override and stop switches to be canceled.

When machining corners, with continuous cutting applied, the slides are - on account of their inertia - unable to follow the path commanded by the control system. Now the tool will round the corner more or less, depending on the feed. If the workpiece requires sharp corners, the control must be specified to slow down at the end of block, wait until the axes come to a halt, and start the next movement only afterwards.

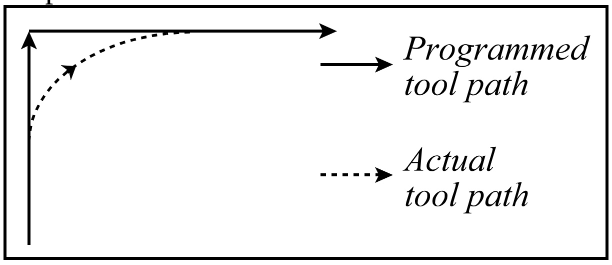


Fig. **6.4-1**

### 6.4.1 Exact Stop (G09)

Not being a modal function, **G09** will be effective only in the block, in which it has been programmed.
At the end of the block, it has been specified, the control will slow down and stop after execution of the interpolation, and will wait for the "in position" signal. Unless that signal arrives in 5 seconds, the control will return a message *1020 POSITION ERROR*.
That function can be used for exact machining sharp corners.

### 6.4.2 Exact Stop Mode (G61)

Modal function, canceled with G62, G63 or G64 command.
The control system will slow down and stop on completion of each interpolation and wait for the "in position" signal. It will start the next interpolation cycle only afterwards. Unless that signal arrives in 5 seconds, the control will return a message *1020 POSITION ERROR*.

### 6.4.3 Continuous Cutting Mode (G64)

Modal function. The control will assume that state after power-on. It will be canceled by codes G61, G62 or G63.
In this mode the movement will not come to a halt on the completion of the interpolation, the slides will not slow down. Instead, the interpolation of the next block will be commenced immediately.
Sharp corners cannot be machined in this mode, because they will be rounded off.

### 6.4.4 Override and Stop Inhibit (Tapping) Mode (G63)

Modal function, canceled by codes G61, G62 or G64.
The feed and spindle override and the feed stop is inhibited in this mode. The override values are taken for 100% (regardless of switch positions). On completion of the interpolation, the system will not slow down, but start next interpolation cycle immediately.
This mode is applicable in various thread cutting and tapping operations.

### 6.4.5 Automatic Corner Override (G62)

Modal function canceled by any of codes G61, G63 or G64.

When inside corners are being machined, higher forces are acting upon the tool before and after the corners. To prevent the overload of the tool and developing vibrations, the control will - when G62 commanded - automatically reduce the feed along before and after an inside corner.

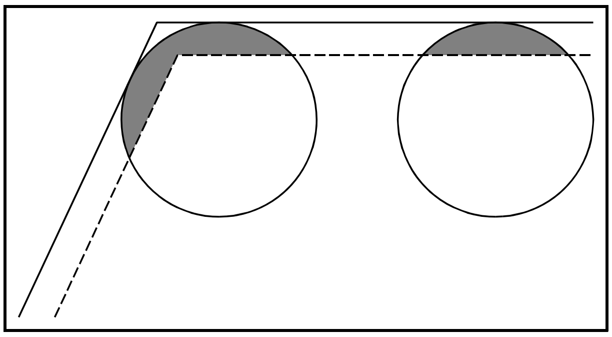


Fig. **6.4.5-1**

The corner override is effective under the following conditions.

– When tool nose radius compensation is on (G41, G42).
– Between blocks G0, G1, G2, G3.
– In movements in the selected plane.
– When the corner is machined inside.
– When the angle of the corner is smaller then a particular angle defined by parameter.
– Over a distance before and after the corner, defined by parameters.

The corner override function will be effective between each the following pairs of blocks: Linear-to-linear, linear-to-circular, circular-to-linear, circular-to-circular ones.

Inside angle Θ can be selected between 1 and 180° by parameter *CORNANGLE*.

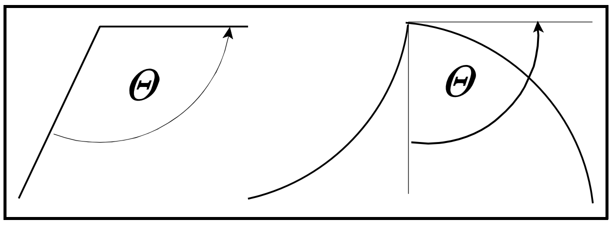


Fig. **6.4.5-2**

Deceleration and acceleration will be commenced at distances $L_l$ and $L_g$ before and after the corner, respectively. In the case of (circles) arcs, distance $L_l$ and $L_g$ will be calculated by the control along the arc. Distances $L_l$ and $L_g$ will be defined in parameters *DECDIST* and *ACCDIST*, respectively.

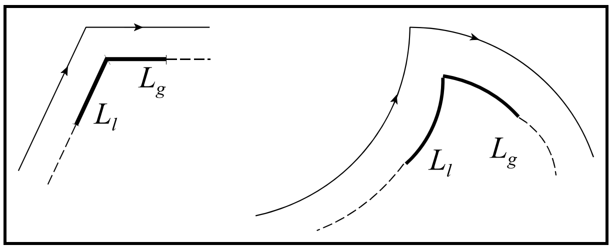


Fig. **6.4.5-3**

The value of override can be selected as a percent in parameter *CORNOVER*. The override will begin to be effective at distance $L_l$ before the corner, and will be effective over distance $L_g$ behind the corner. The values of feed override and corner override will be taken into account together by the control:

F* feed override * corner override.

Write G09 in the particular block to program an exact stop in state G62.

### 6.4.6 Internal Circular Cutting Override

With the tool nose radius compensation on (G41, G42), the control will automatically reduce the feed in machining the inside surface of an arc so that the programmed feed will be effective along the cutting radius. The feed in the center of the tool radius is

$$F_c = \frac{R_c}{R} F$$

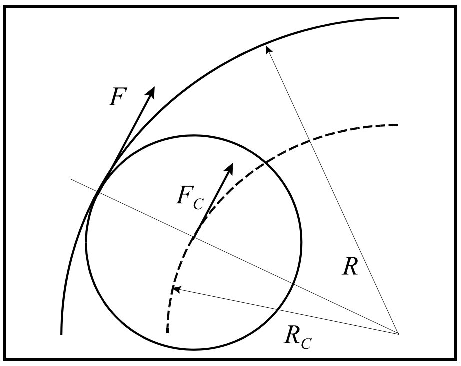


Fig. **6.4.6-1**

where $F_c$ is the (corrected) feed of the tool-radius center
R is the programmed radius of circle
$R_c$ is the corrected radius of circle
F is the programmed feed.

The lower limit of automatic feed reduction is set by parameter *CIRCOVER*, in which the minimum override can be specified as a percent. The override for the circle radius is multiplied by the values of feed and corner override before it is issued.

## 6.5 Automatic Deceleration at Corners

Rapid traverse positioning G0 always decelerates to 0 feed at block end position and the execution of the next movement block starts with acceleration from 0 feed. Between feed blocks (G1, G2, G3) at block end position the control only decelerates in case of the right parameter setting, change of feedrate and in case it detects direction change, i.e. “corner”. If there is no path break, i.e. strong direction change, deceleration is senseless and detrimental.

There are two reasons for detecting feed changes (corners) and at the same time for decelerating feed:

– Feed changes per axis deriving from abrupt direction changes may be so high, that without deceleration the drives are not able to follow the path without swing on the account of accuracy, as well as taxes the machine tool mechanically.
– Abrupt path direction change means corner and if the corner is to be set sharp during machining, it also needs to be decelerated. The more the feed is decelerated, the sharper the corner becomes.

If no deceleration is executed at the corner in two subsequent N1, N2 blocks, feed differences ($\Delta F_x$, $\Delta F_z$) occur along the axes, which results in the tool rounding the real corner.

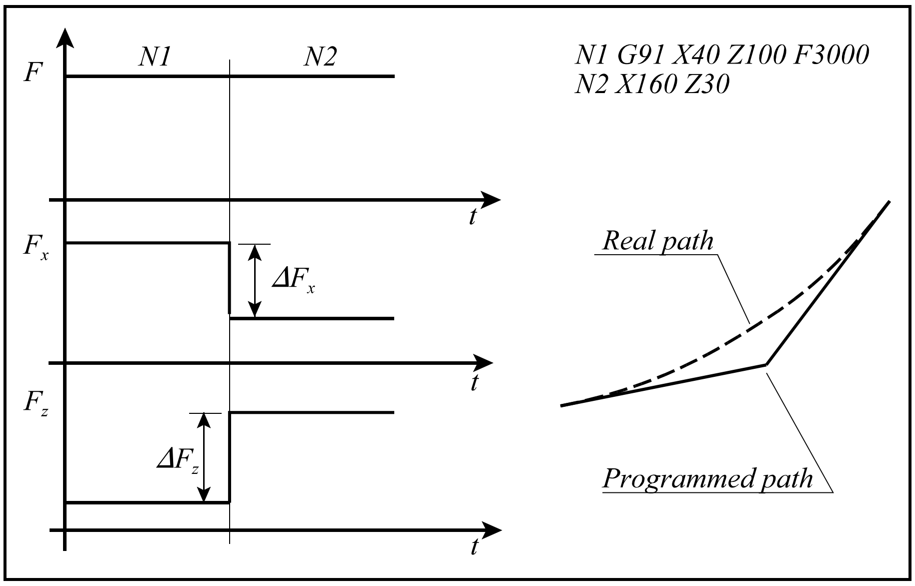


Fig. **6.5-1**

*For the corner deceleration function to operate, parameter 2501 CDEN must be set to 1.*

The control can execute corner detection in two ways: by monitoring the change of path direction angle or the change of feed components per axis. The method to be used can be chosen with the help of parameters.

Deceleration at corners by monitoring the change of path direction angle

In case of parameter settings *2501 CDEN*=1 and *2502 FEEDDIF*=0 deceleration is executed by monitoring the change of path direction angle. This setting also operates in states G94 (feed per minute) and G95 (feed per revolution).

☞ *Warning: Since in case of turning machines mostly feed per revolution is programmed, this setting is needed here.*

If angle α exceeds the value enabled at the parameter at the meeting point of blocks N1, N2 the control decelerates feed to value $F_c$.

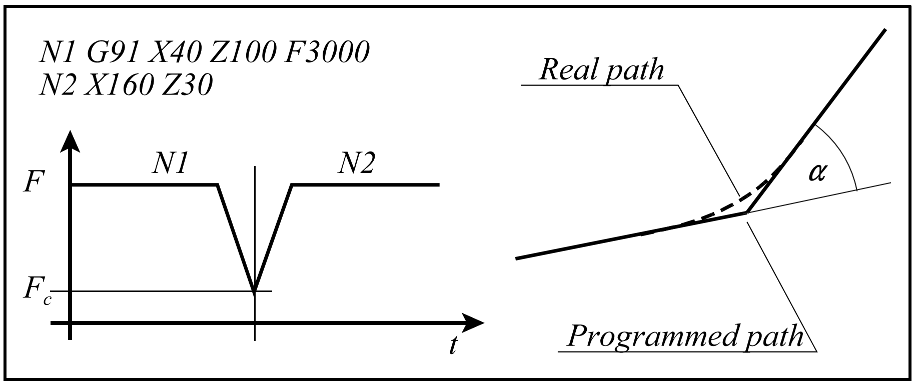


Fig. **6.5-2**

The value of the critical angle can be set at parameter *2511 CRITICAN* in degree. Value of parameter *2512 FEEDCORN* defines the feed to which the control must decelerate when the critical angle is exceeded: $F_c$=*FEEDCORN*.

Deceleration at corners by monitoring the cange of feed components per axis

If *2501 CDEN*=1 and *2502 FEEDDIF*=1, deceleration is executed by monitoring the change of feed components. This setting only operates in state G94 (feed per minute).

If feed is decelerated at the meeting point of blocks N1, N2, so that on neither axis does the feed change exceed the critical feed difference ($\Delta F_{xmax}$, $\Delta F_{zmax}$) enabled for the axis at parameter, then the tool sharpens the corner in function of the critical feed. In order to get the feed at the corner point, the critical feed values must be divided for each axis by the value of the resulting feed changes and the minimal value must be multiplied by the programmed feed. It decelerates to the resulting feed $F_c$ at the corner:

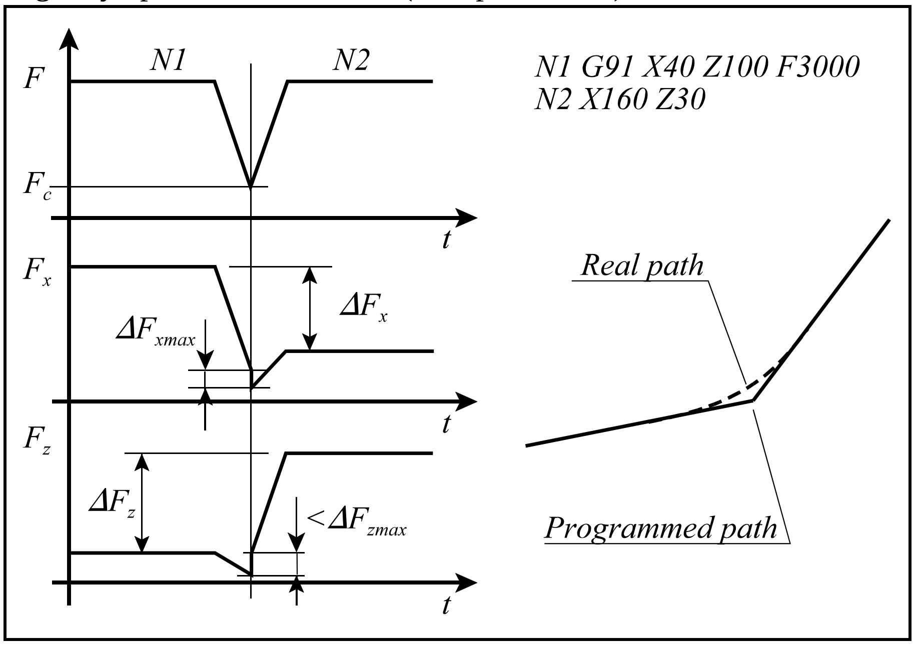


Fig. 6.5-3

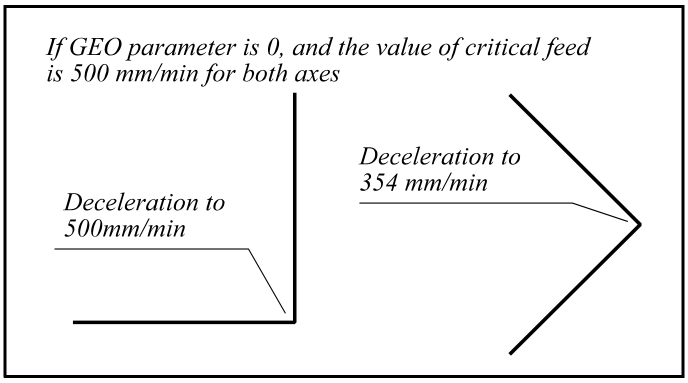
$$F_c = \min\left\{\frac{\Delta F_{x\max}}{\Delta F_x}, \frac{\Delta F_{z\max}}{\Delta F_z}, \ldots\right\} \times F$$

where:

$\Delta F_{xmax}$, $\Delta F_{zmax, \ldots}$: the respective parameter *252n CRITFDIFn* set for axes X, Z, ...,
$\Delta F_x$, $\Delta F_z$, ...: feed change on axes X, Z, ....

The value of decreased feed depends on the geometric position of the corner. See the following example:

If a 90˚ corner is passed in directions parallel to the axes and the critical feed is 500mm/min on both axes, the feed must be decelerated to this speed at the corner.
If however the legs of rectangular corner enclose 45˚ with the axes, it must be decelerated to 354 mm/min.
In case of parameter setting 2503 *GEO* =0 the control operates as above.
Thus the feed will always be the highest possible.



If GEO parameter is 0, and the value of critical feed is 500 mm/min for both axes
Deceleration to 354 mm/min
Deceleration to 500mm/min

Fig. 6.5-5

In case of parameter setting *2503 GEO* =1 the control starts from the worst case (45˚) and operates with the feed valid in the case of 45˚ independent of the geometric position of angle legs. This can result in at most 30% feed decrease.

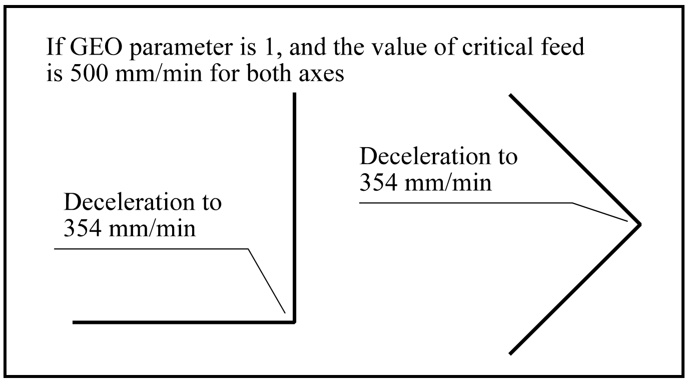

If GEO parameter is 1, and the value of critical feed is 500 mm/min for both axes
Deceleration to 354 mm/min
Deceleration to 354 mm/min

Fig. **6.5-6**

☞ ***Warning:***

*Automatic feed deceleration at corners differs from exact stop function (G9, G61). In the latter case the control always decelerates to 0 in all block end positions and waits for signal "in position".*

*It also differs from automatic feed decrease at internal corners (G62), which can only be affected in states G41, G42. In this case the feed value is already decreased to the distance set at parameter before the corner.*

## 6.6 Limiting Accelerations in Normal Direction along the Path in Case of Circular Arcs

The control keeps the feed at constant value along the path tangent (in tangential direction) in the course of machining. Conclusively there are no acceleration components in tangential direction. This is not the case in normal direction (perpendicular to path or speed). The normal direction acceleration components on axes may exceed the value enabled for the axis. In order to avoid this, the speed along the path must be limited in proportion of the path curvature.

When machining circular arcs the feed size F is limited on the basis of the following relation

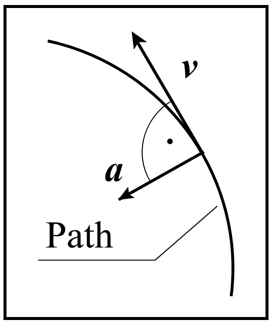


Fig. **6.6-1**

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

where:

*a*: the lower of the acceleration values *(parameters 470n ACCn)* set for axes involved in circular interpolation,

*r*: circle radius.

Circular interpolation is started at the speed calculated.

Independent of the above relation, the speed is not decreased lower than the feed set at parameter *2513 CIRCFMIN.*

☞ ***Warning:***

*This function differs from automatic feed decrease in states G41, G42 when machining inner arches of circles.*

# 7 The Dwell (G04)

The

    (G94) **G04** P....

command will program the dwell in seconds.
The range of P is 0.001 to 99999.999 seconds.

The

    (G95) **G04** P....

command will program the dwell in terms of spindle revolutions.
The range of P is 0.001 to 99999.999 revolutions.
Depending on parameter *SECOND*, the delay may refer always to seconds as well, irrespective of the states of G94, G95.
The dwell implies invariably the programmed delay of the execution of the next block. It is a non-modal function.
During dwell in status field 5 indicating interpolation status the message DWL will appear on screen to draw the attention of operator why the machine is halted.

# 8 The Reference Point

The reference point is a distinguished position on the machine-tool, to which the control can easily return. The location of the reference point can be defined as a parameter in the coordinate system of the machine. Work coordinate system can be measured and absolute positioning can be done after reference point return. The parametric overtravel positions and the stroke check function are only effective after reference-point return.

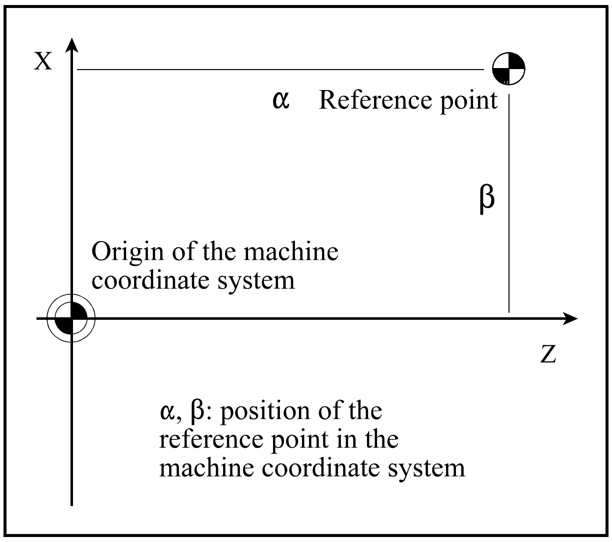

X
α
Reference point
β
Origin of the machine coordinate system
Z
α, β: position of the reference point in the machine coordinate system

Fig. **8-1**

## 8.1 Automatic Reference Point Return (G28)

The instruction

**G28** v

will return the axes defined by vector v to the reference point. The movements consist of two parts.
First it will move with linear interpolation in rapid traverse to the intermediate coordinates defined by vector v. The specified coordinates may be absolute or incremental values. The movement is performed invariably in the current coordinate system .
When the end point of linear movement is reached, the tool nose radius compensation vector is deleted.
The coordinates of the intermediate point will be stored for axes defined by vector v.
In the second stage it will move from the intermediate point to the reference-point simultaneously in each axis defined by vector v. The reference-point return is carried out by non-linear movement at a speed defined for each axis. Afterwards, similar to the manual return, the position will be assumed in the manner defined by parameters.
This is a non-modal code.

☞ *Notes*:
– Unless there is a valid reference point, incremental values must be assigned to intermediate coordinates v in command G28.
– Programmed in block G28, intermediate coordinates v will be stored until power-off. In other words, the intermediate value defined in a previous command G28 will continue to be effect for the coordinates that have not been assigned values in the instantaneous (current) command G28. For example:

| | |
|---|---|
| G28 X100 | intermediate point: X=100, Z=0 |
| G28 Z200 | intermediate point: X=100, Z=200 |

## 8.2 Automatic Return to Reference Points 2, 3, 4 (G30)

Series of instructions

**G30** v P

will send the axes of coordinates defined at the addresses of vector v to the reference point defined at address P.

P1=reference point 1
P2=reference point 2
P3=reference point 3
P4=reference point 4

The reference points are special positions defined by parameters (REFPOS1, ..., REFPOS4) in the coordinate system of the machine-tool, used for change positions, e.g., positions of tool change or palette change. The first reference point is invariably the position of the machine's reference point, i.e., the point to which the control moves when returning to the reference point.
The instruction is only applicable after the machine's reference point has been returned.
The movement consists of two parts. First it will move by a linear motion to the intermediate coordinates defined by vector v, with rapid traverse. The specified coordinates may be absolute or incremental values. The movement is carried out invariably in the current coordinate system. When the end point of linear movement is reached, the tool nose radius compensation vector will be deleted. The coordinates of the intermediate point will be stored in the current coordinate system for the axes defined by vector v. Stored in this way, the coordinates will overwrite those stored in instruction G28.
In the second phase, the axes defined by vector v will move with rapid traverse from the intermediate point to the reference point selected at address P.
The reference point is returned by disregarding the compensation vectors (length, offset, 3 dimensional offsets) they need not be deleted before instruction G30 is issued but they will be implemented by the control when further movements are being programmed. The tool nose radius compensation is re-established automatically in the first movement block.
A non-modal code.

## 8.3 Automatic Return from the Reference Point (G29)

Instruction

**G29** v

will return the control from the reference point along the axes defined in vector v. Following G28 and G30, command G29 will be executed in the same manner. The return is accomplished in two stages.
In the first stage it will move from the reference point to the intermediate point recorded during the execution of instruction G28 or G30, in the axes defined by vector v. The coordinates of the intermediate point are modal, in other words, the control will take the previous values into account if reference is made to an axis, to which no coordinate has been transferred in block G28 or G30 preceding G29. It will move to the intermediate point by taking into account the tool length, tool offset and 3-dimensional tool radius compensations.
The coordinates of the intermediary point are effective invariably in the coordinate system of the current workpiece. Accordingly if, e.g., a change of workpiece coordinate system has been programmed after reference point return and before instruction G29, the intermediate point will be taken into account in the new coordinate system.
In the second phase it will move from the intermediate point to the point v defined in instruction G29. If coordinate v has an incremental value, the displacement will be measured from the inter-

mediate point.
When the tool nose radius compensation is set up, it will move to the end point by taking into account the compensation vector.
A non-modal code.
An example of using G30 and G29:

```
...
G90
...
G30 P1 X200 Z500
G29 X150 Z700
...
...
```

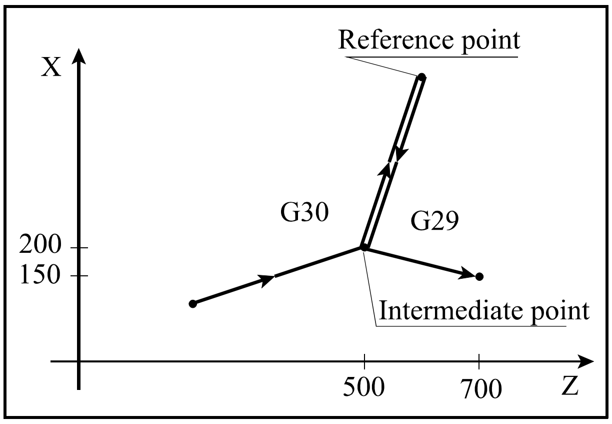


Fig. **8.3-1**

# 9 Coordinate Systems, Plane Selection

The position, to which the tool is to be moved, is specified with coordinate data in the program. When 2 axes are available (X, Z), the position of the tool is expressed by two coordinate data X____ Z____ :

The tool position is expressed by as many different coordinate data as is the number of axes on the machine. The coordinate data refer invariably to a given coordinate system.

The control will differentiate three different coordinate systems.

1. the machine coordinate system,
2. the workpiece's coordinate system,
3. the local coordinate system.

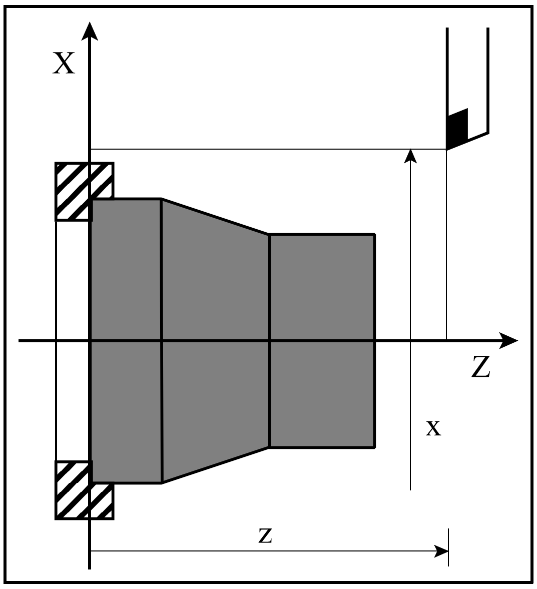
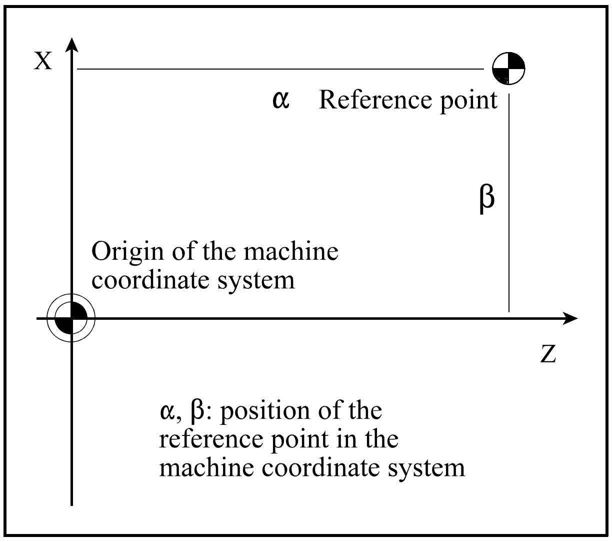


Fig. 9-1

## 9.1 The Machine Coordinate System

The machine zero point, i.e., the origin of the machine coordinate system, is a point on the given machine-tool, that is usually defined by the machine tool builder. The control will define the machine coordinate system at the time of returning to the reference point.

Once the machine coordinate system has been defined, it will not be altered by the change of the work coordinate system (G54 ... G59) or by other coordinate transformation (G52, G92), only by a power-off of the control system.


X
α
Reference point
β
Origin of the machine coordinate system
Z
α, β: position of the reference point in the machine coordinate system
Fig. 9.1-1

### 9.1.1 Setting the Machine Coordinate System

After a reference point return, the machine coordinate system can be set in parameters. The distance of the reference point, calculated from the origin of the machine coordinate system, has to be written for the parameter.

### 9.1.2 Positioning in the Machine Coordinate System (G53)

Instruction

**G53** v

will move the tool to the position of v coordinate in the machine coordinate system.

– Regardless of states G90, G91, coordinates v are always treated as absolute coordinates,
– operator I is ineffective when put behind the address of a coordinate,
– similar to instruction G00, the movements are performed in rapid traverse,
– the positioning is carried out invariably with the selected tool length compensations taken into account.

A G53 instruction can be executed after a reference point return only. G53 is a one-shot command effective in the block only, where it has been specified.

## 9.2 Work Coordinate Systems

The coordinate system applied in cutting the workpiece is referred to as the "work coordinate system". Six different coordinate systems can be defined for the workpiece in the control.

### 9.2.1 Setting the Work Coordinate Systems

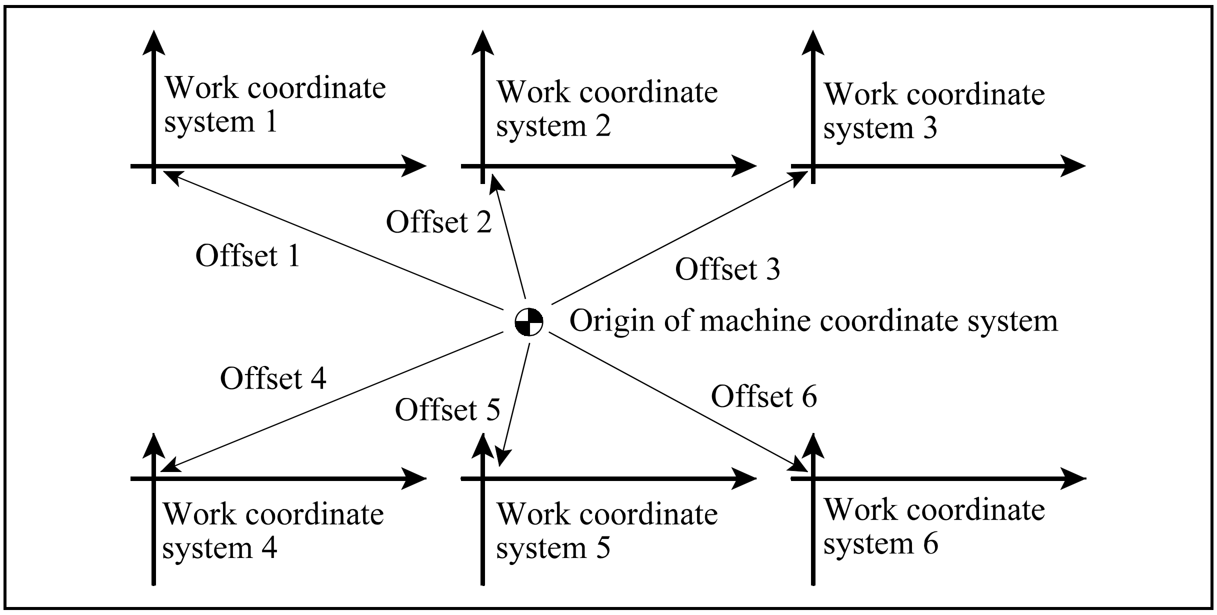


Fig. **9.2.1-1**

In setting mode the locations of the various work coordinate systems can be established in the machine coordinate system, and the necessary offsets can be made.

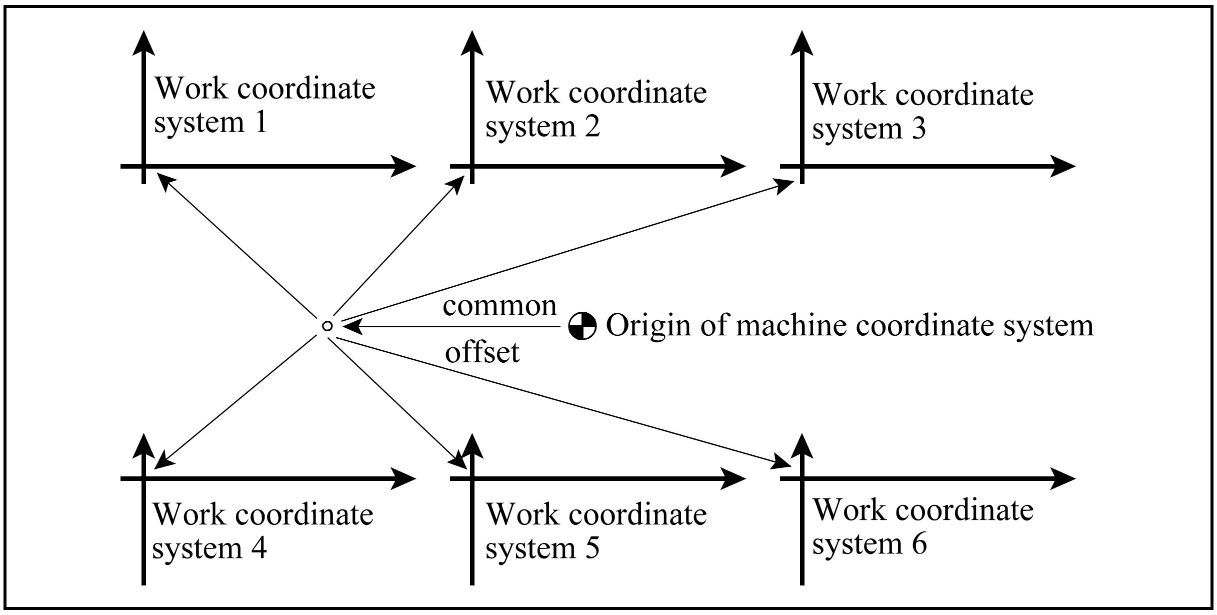


Fig. **9.2.1-2**

Furthermore, all work coordinate system can be offset with a common value. It can also be entered in setting mode.

### 9.2.2 Selecting the Work Coordinate System

The various work coordinate system can be selected with instructions G54...G59.

**G54**........work coordinate system 1
**G55**........work coordinate system 2
**G56**........work coordinate system 3
**G57**........work coordinate system 4
**G58**........work coordinate system 5
**G59**........work coordinate system 6

They are modal functions. Their selection before a reference point return is ineffective. After a reference point return, work coordinate system 1 (G54) will be selected.

The absolute coordinate data of the interpolation blocks will be taken into account by the control in the current work coordinate system.

For example, the instruction
```
G56 G90 G00 X80 Z60
```
will move the system to point X=80, Z=60 of work coordinate system 3.

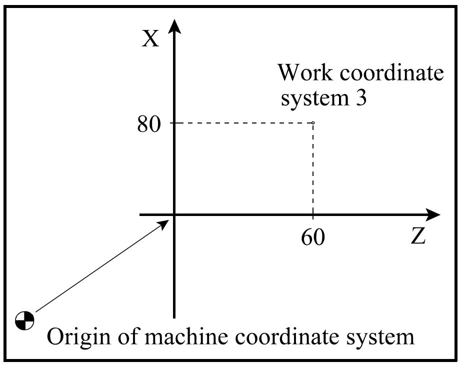


Fig. **9.2.2-1**

After a change of the work coordinate system, the tool position will be displayed in the new coordinate system. For instance, there are two workpieces on the table. The first work coordinate system (G54) has been assigned to zero point of one of the workpieces, which has an offset of X=260, Z=80 (calculated in the machine coordinate system). The second work coordinate system (G55) has been assigned to the zero point of the other workpiece, which has an offset of X=140, Z=180 (calculated in the machine coordinate system). The tool position is X'=140, Z'=90 in X', Z's' coordinate system (G54). As a result of instruction G55, the tool position will be interpreted in the X", Z" coordinate system (X"=260, Z"=–50).

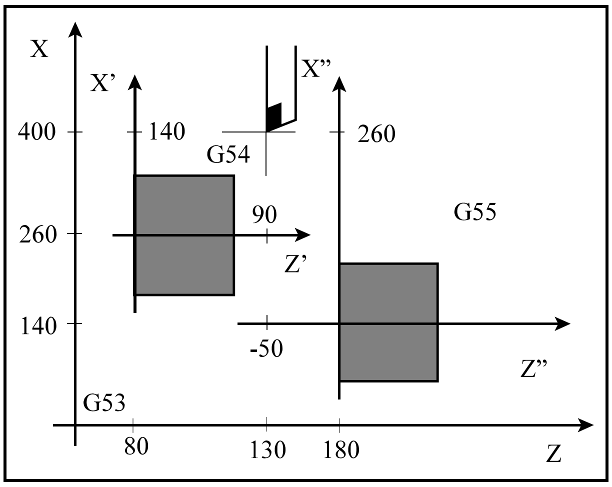


Fig. 9.2.2-2

### 9.2.3 Programmed Setting of the Work Zero Point Offset

It is also programable to set the work coordinate system and the common offset thereof with program instructions.
This is accomplished with instruction

**G10** v **L2** Pp

where

p = 0   sets the common offset,
p = 1...6 selecting work coordinate system 1.- 6.
v (X, Z, ...) = offset for each axis.

The coordinate data are entered invariably as rectangular (Cartesian) absolute values. G10 is a one-shot (non-modal) instruction.

## 9.2.4 Creating a New Work Coordinate System (G92)

Instruction

**G92** v

will establish a new work coordinate system in such a way that coordinate point v of the new system will be a selected point - e.g. the tool's tip (if a length compensation is programmed) or the base point of the tool holder (in lack of a length compensation). Afterwards any additional absolute command will refer to that new work coordinate system, and the positions will also be displayed in that coordinate system. The coordinates specified in command G92 will always be interpreted as rectangular absolute values.

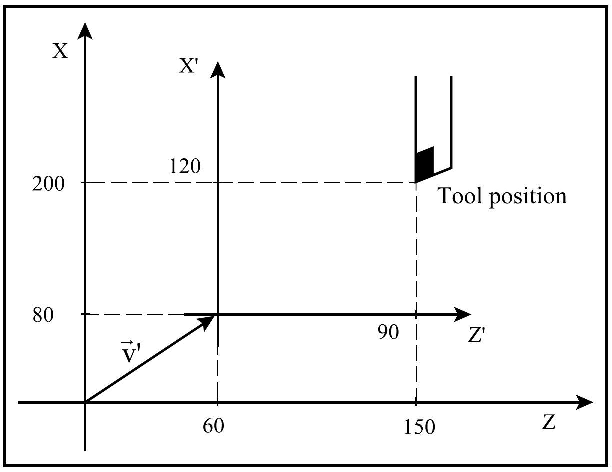


Fig. **9.2.4-1**

If, e.g., the tool is at a point of X=200 , Z=150 coordinates, in the actual (current) X, Z work coordinate system, instruction

```
G92 X120 Z90
```

will create a new X', Z' coordinate system, in which the tool will be set to the point of X'=120, Z'=90 coordinates. The axial components of offset vector v' between coordinate systems X, Z and X', Z' are

$v'_x$=200-80=120,

and

$v'_z$=150–90=60.

Command G92 will prevail in each of the six work coordinate systems, i.e., an offset v calculated for one of them will be taken into account in the rest, too.

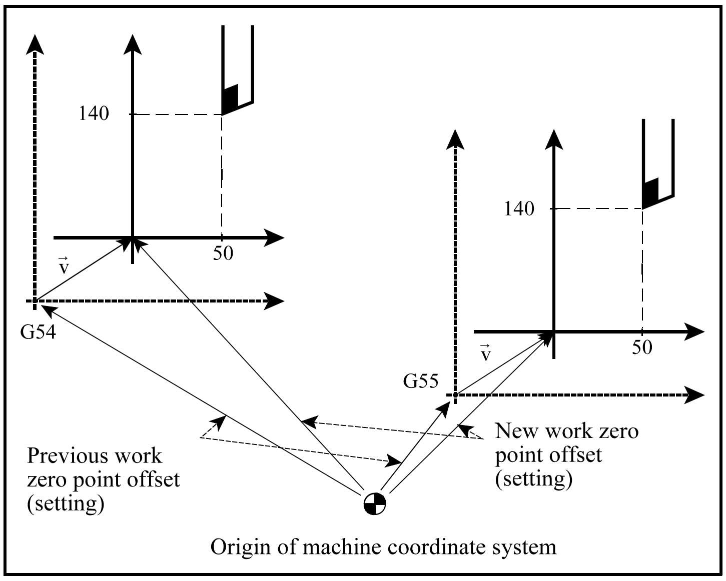

140
50
v
G54
140
50
v
G55
Previous work zero point offset (setting)
New work zero point offset (setting)
Origin of machine coordinate system

Fig. **9.2.4-2**

☞ *Notes*:

– The offset of the work coordinate system set with instruction G92 will be deleted by execution of "end of program" instructions (M2, M30) and by resetting the program.

– Instruction G92 will delete the offsets of the local coordinate system (programmed with instruction G52) on the axes included in the instruction.

## 9.3 Local Coordinate System

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

Instruction

**G52** v

will create a local coordinate system.

– If coordinate v is specified as an absolute value, the origin of the local coordinate system will coincide with the point v in the work coordinate system.

– When specified as an incremental value, the origin of the local coordinate system will be shifted with v offset (provided a local coordinate system has been defined previously, or else the offset is produced with respect to the origin of the work coordinate system).

Henceforth any movement command specified in absolute coordinates will be executed in the new coordinate system. The positions are also displayed in the new coordinate system. The values of coordinates v will be treated invariably as Cartesian coordinates.

If, e.g., the tool is at point X=200, Z=150 coordinates in the current X, Z work coordinate system, instruction

```
G90 G52 X80 Z60
```

will create a new local X', Z' coordinate system, in which the coordinates of tool will be X'=120, Z'=90. Instruction G52 is used for defining the axial components of offset vector v' between the X, Z and X', Z' coordinate systems ($v'_x$=80, $v'_z$=60).

Now one of two different procedures may be adopted in order to transfer the local coordinate system to the point of X", Z" position.

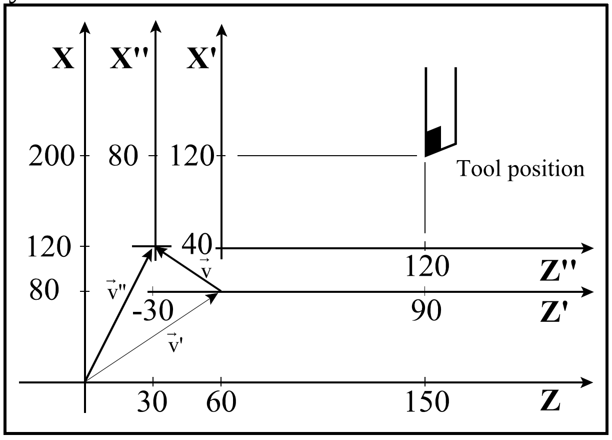


Fig. **9.3-1**

– With an absolute data specification: Instruction (G90) G52 X120 Z30 will move the origin of the X", Z" coordinate system to point X=120, Z=30 in the X, Z work coordinate system. The components of vector v" will be produced by the specification of $v''_x$=120, $v''_z$=30.

– With an incremental data specification: Instruction G91 G52 X40 Z–30 will move the origin of the X", Z" coordinate system to the point of X'=-40, Z'=30 coordinates in the X', Z' coordinate system. The components of vector v will be produced by the specification of $v_x$=40, $v_z$=30. Indicating the location of the new local coordinate system in the X, Z work coordinate system vector v"=v'+v. Its components:

$v''_x$=80+40=120, $v''_z$=60+(–30)=30.

The tool position in the X", Z" coordinate system will be X"=80, Z"=120.

Instruction

**G90 G52 v0**

will delete the offset in on coordinates specified in v.

The local coordinate system will be offset in each work coordinate system.

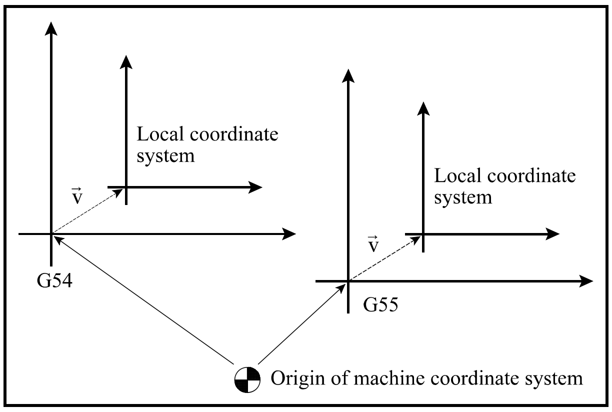


Fig. **9.3-2**

Programming instruction G92 will delete the offsets produced by instruction G52 on the axes specified inG92 - as if command G52 v0 had been issued.

Whenever the tool is at point of X=240, Z=200 coordinates in the X, Z work coordinate system, instruction

```
G52 X80 Z60
```

will shift its position to X'=160, Z'=140 in the X', Z' local coordinate system.

Now instruction

```
G92 X80 Z110
```

will establish the tool position to X"=80, Z"=110 in the new X", Z" work coordinate system. Thus the X', Z' local coordinate system will be deleted by command G92 as if command G52 X0 Z0 had been issued.

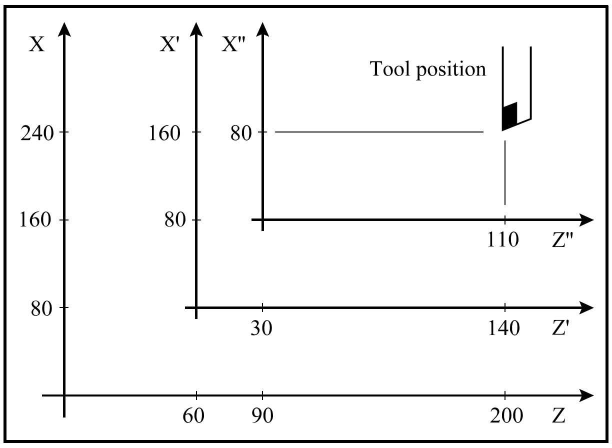


Fig. **9.3-3**

☞ *Note*:

– The offset of the local coordinate system will be deleted by execution of commands M2, M30 and/or by resetting the program.

## 9.4 Plane Selection (G17, G18, G19)

The plane in which

– circular interpolation,
– tool nose radius compensation,
– positioning of drilling cycles

will be performed can be selected with the following G codes:

**G17**............$X_pY_p$ plane
**G18**............$Z_pX_p$ plane
**G19**............$Y_pZ_p$ plane,

where

$X_p$=X or an axis parallel to X,
$Y_p$=Y or an axis parallel to Y,
$Z_p$=Z or an axis parallel to Z.

The selected plane is referred to as "main plane".

The particular one of the parallel axes will be selected (by instruction G17, G18 or G19) depending on the axis addresses programmed in a given block:

When X and U, Y and V, Z and W are parallel axes:

The XY plane will be selected by G17 X_Y_,
the XV plane will be selected by G17 X_V_,
the UV plane will be selected by G17 U_V_,
the XW plane will be selected by G18 X_W_,
the YZ plane will be selected by G19 Y_Z_,
the VZ plane will be selected by G19 V_Z_.

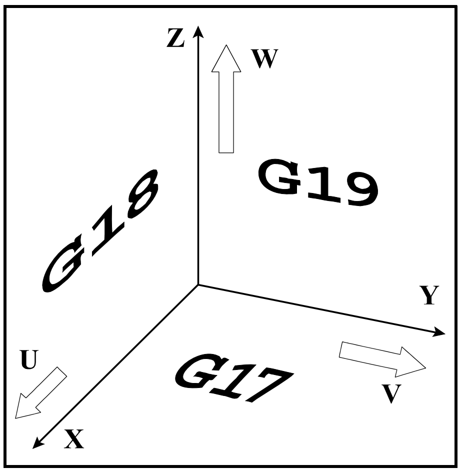


Fig. **9.4-1**

Unless G17, G18, G19 is specified in a block, the selected plane remains unchanged:

G17 X____ Y____ plane XY
U____ Y____ plane XY remains.

Unless there is an axis address specified in the G17, G18, G19 block, the control will consider the basic axes:

    The XY plane will be selected by G17,
    the XY plane will be selected by G17 X,
    the UY plane will be selected by G17 U,
    the XV plane will be selected by G17 V,
    the ZX plane will be selected by G18,
    the WX plane will be selected by G18 W.

The selected plane is unaffected by the movement command:

```
      (G90) G17 G00 Z100
```

will select the XY plane, moving the Z axis to the point of coordinate 100.
After power-on, the default plane (G17 or G18) is specified according to the parameter group *CODES*.
The main plane can be selected more times in the same program.
Address U, V, W can be selected as a parallel in parameters.

# 10 The Spindle Function

## 10.1 Spindle Speed Command (Code S)

With a number of max. five digits written at address **S**, the NC will give a code to the PLC. Depending on the design of the given machine-tool, the PLC may interpret address S as a code or as a data of revs/minute.
When a movement command and a spindle speed (S) are programmed in a given block, function S will be issued during or after the motion command. The machine tool builder will define the way of execution.
The speeds specified at address S are modal values. At the time of power-on, the control will assume value S0. The spindle speed has a minimum and a maximum limit in each gear ratio range. They are defined by the machine-tool builder in parameters and the control does not let the speed outside of this range.

## 10.2 Programming of Constant Surface Speed Control

Constant surface speed control function can only be used in case of infinitely variable speed main drive. In this case the control can change the spindle speed so that the tool speed is constant relative to the surface of the workpiece and is equal to the programmed value.
The constant surface speed must be specified in function of the input unit on the basis of the table below:

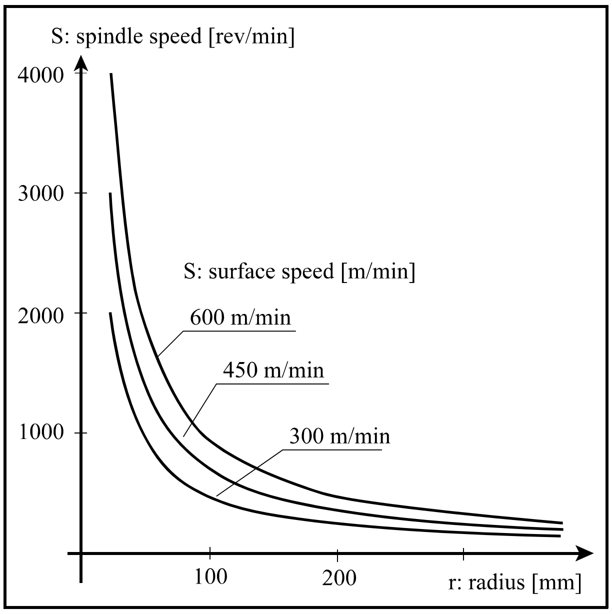


Fig. **10.2-1**

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

### 10.2.1 Constant Surface Speed Control Command (G96, G97)

Command

**G96** S

switches constant surface speed control function on. The constant surface speed must be specified at address S in the unit of measure given in the above table.

Command

**G97** S

cancels constant surface speed control. The desired spindle speed can be specified at address S (in revs/min).

– In order to calculate constant surface speed the coordinate system must be set so that its zero point coincides with the rotation axis.

– Constant surface speed control is effective only after the spindle is started by means of M3 or M4.

– The  value is modal even after its calculation has been canceled with the help of command G97. After power on the default constant surface speed is determined by parameter CTSURFSP.

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

– Constant surface speed calculation is also effective in state G94 (feed/min).

– If the constant surface speed control is canceled by means of command G97 and a new spindle speed is not specified the last spindle speed gained in state G96 remains in effect.

```
G96 S100    (100m/min or 100 feet/min)
      .
      .
      .
G97         (Revolution belonging to resulting diameter X)
```

– In case of rapid traverse positioning (block G00) the constant surface speed is not calculated continuously but the revolution belonging to the end-position will be calculated. This is needed for the spindle speed to avoid unnecessary changes.

– In order to calculate constant surface speed the zero point of the axis, on the basis the spindle speed is changed, must be set to the spindle rotation axis.

### 10.2.2 Constant Surface Speed Clamp (G92)

With the help of command

**G92** S

the highest spindle speed enabled in case of constant surface speed control can be set. During constant surface speed calculation the control clamps spindle speed to this value. In this case the unit of S is rpm.

– After power on as well as if value S has not been clamped by means of command G92 the top limit of spindle speed in case of constant surface speed control is the maximum value enabled for the given gear range.

– The maximum revolution value is modal until
a new one is programmed,
contol system runs to the end of program,
or until mode change.

### 10.2.3 Selecting an Axis for Constant Surface Speed Control

The axis, which position the constant surface speed is calculated from, is selected by parameter 1182 AXIS. The logic axis number must be written at the parameter.
If other than the selected axis is to be used, the axis from which the constant surface speed is to be calculated can be specified by means of command

**G96** P.

Interpretation of address P:

P1: X, P2: Y, P3: Z,
P4: U, P5: V, P6: W,
P7: A, P8: B, P9: C

– The value set at address P is modal. After power on the control activates constant surface speed control to the axis set at parameter AXIS.

## 10.3 Spindle Position Feedback

In normal machining the NC will issue a speed command to the power amplifier of the spindle, proportional to the programmed speed (value specified at address S). Now this amplifier will be working in speed-control mode.
Some technological tasks may, however, require the spindle to be brought to a particular angular position. This is referred to as spindle positioning or indexing.
Prior to positioning, the NC will set the power amplifier of the spindle to position-controlled mode. In practice this means that the NC will not issue a speed command proportional to code S any more, instead, it will measure the position of the spindle by the use of an encoder mounted on the spindle, and will issue a command to the servo amplifier in accordance with the desired angular displacement (similar to the rest of controlled axes). This is the position feedback.
To be able to position the spindle on a particular machine, an encoder has to be mounted on the spindle and the power amplifier of the spindle must be capable of operation in position feedback mode as well.

## 10.4 Oriented Spindle Stop

The "spindle orientation" or the "oriented spindle stop" refers to the function of stopping the spindle in a particular angular position. This may be necessary, e.g., for an automatic tool change or for the execution of some drilling cycles. The possibility of orientation on a particular machine must be specified by parameter *ORIENT1* in parameters. The command of spindle orientation is issued by function M19, but it may also be produced by some other function depending on the particular machine-tool. The orientation may be carried out in one of two different ways.
If the spindle cannot be used in position control mode, the orientation is feasible by turning the spindle to a proximity switch mounted on the machine.
If the spindle can be used in position control mode, command M19 will cause the control to return to the zero pulse of the spindle encoder. The control will automatically close the position control loop.

## 10.5 Spindle Positioning (Indexing)

A spindle positioning is only feasible after the spindle position control loop has been closed after orientation. Accordingly, this function is used for closing the loop. The loop will be opened by rotation command M3 or M4.
If the value of parameter *INDEX1*=1 (indicating that the main drive position control loop can be

closed) and the value of parameter *INDEX_C1*=0, the spindle indexing will be performed by function M.
Under such conditions function M from the threshold value set on parameter *M_NUMB1* to *M_NUMB1*+360 will be interpreted as a spindle indexing commands, i.e., the threshold number will be subtracted from the programmed value of M, and the number obtained will be treated as an incremental displacement specified in degrees.
Thus, if *M_NUMB1*=100, command M160 means that the spindle must be turned by 160-100=60 degrees from its current position. The direction of rotation is selected by parameter *CDIRS1*, its rate is selected by parameter *RAPIDS1*.

## 10.6 Spindle Speed Fluctuation Detection (G25, G26)

Command

**G26**

enables spindle speed fluctuation detection, while command

**G25**

cancels it. After power-on or RESET the control is set to state G26, i.e., spindle speed fluctuation detection is on. This function signals abnormalities occurring in the course of spindle rotation, as the result of which, e.g., spindle seizure can be avoided.
The speed fluctuation detection is influenced by 4 parameters. These parameters can be overwritten from a program with addresses following command G26. The overwritten parameters are kept upon power-off. The parameters are overwritten as the effect of command

**G26** Pp Qq Rr Dd.

The below table contains the parameter interpretations:

| name | parameter | meaning | unit | value limit |
|---|---|---|---|---|
| p | 5001 TIME | time from the issue of a new spindle speed command to the start of checking | 100 msec | 65535 |
| q | 5002 SCERR | tolerance of the specified spindle speed | % | 1-50 |
| r | 5003 FLUCT% | allowable amount of spindle speed fluctuation in the percentage of programmed speed | % | 1-50 |
| d | 5004 FLUCTW | spindle speed fluctuation in absolute value | revs/min | 65535 |

The process of speed fluctuation detection is as follows.

## Start of Spindle Speed Fluctuation Detection

As the effect of new rotation speed the detection is suspended by the control. The speed fluctuation detection starts when

- the current spindle speed reaches the specified spindle speed within the tolerance limit determined by value "q", or

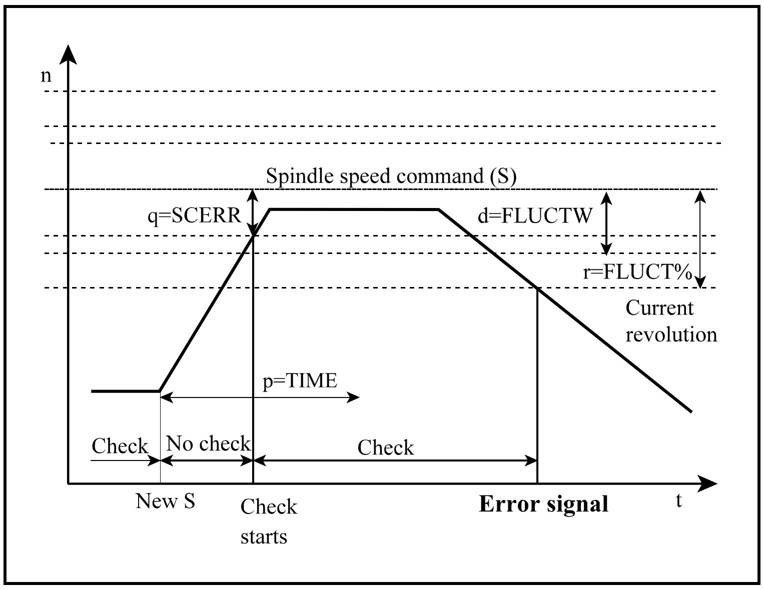


Fig. **10.6**-1

- the current spindle speed has not reached the specified spindle speed within the tolerance limit determined by value "q", but time determined by value "p" has elapsed from the command .

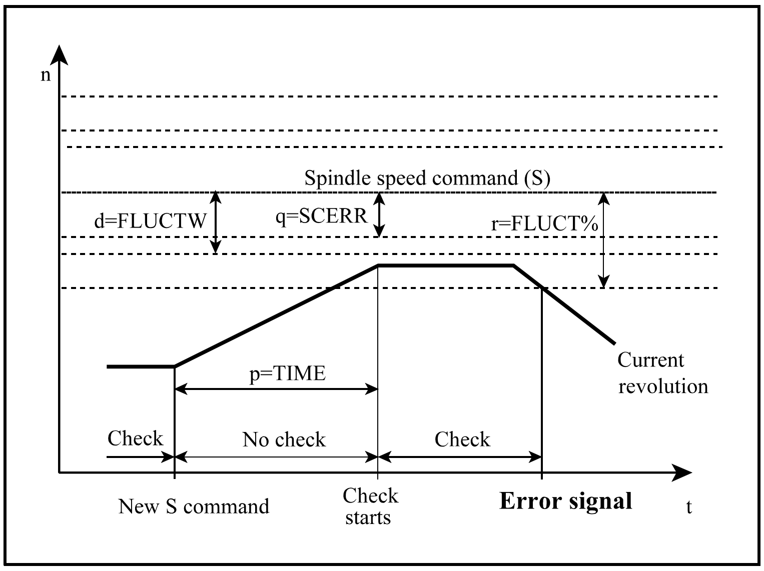


Fig. **10.6**-2

Detecting Error

In the course of detection the control sends error message in case the deviation between current and specified spindle speed exceeds

- the tolerance limit specified by value "r" in percent of the command value and
- also the absolute tolerance limit specified by value "d"

When the current speed has exceeded both tolerance limits, the NC sets flag I656 to PLC. The speed range, in which the NC issues alarm, can be seen on the 3rd figure. If the specified spindle speed is under value "S" apparent in the figure, the NC issues alarm, provided the current speed is 0 revs/min for more than 1 second.

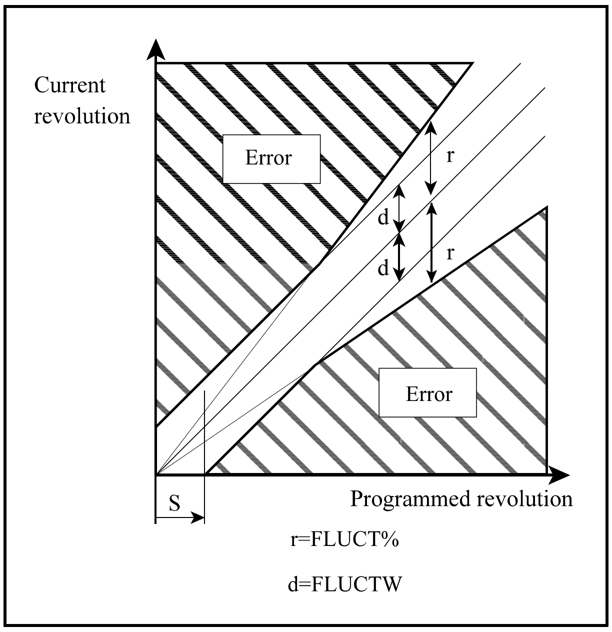


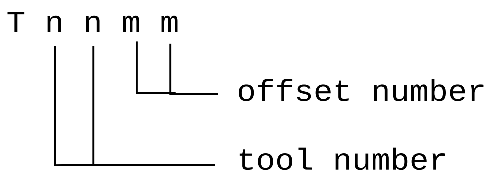
Fig. **10.6-3**

– The spindle speed fluctuation detection is effective only if the spindle is mounted with encoder.
– The specified spindle speed, according to which the current spindle speed is detected is calculated by taking the override, the revolution range limits and the programmed maximum revolution (G92 S_) in constant surface speed calculation (G96) into account.
– The spindle speed fluctuation detection is effective only in case of G26 and rotating spindle (state M3 or M4).
– Command G26 must be programmed in single block.

# 11 Tool Function

The first two digits of the number written at address T form the tool number, while the second two digits form the offset number.

Interpretation of the code written at address T:



Meaning of command T1236: Activate tool No. 12 and use offset number 36.

– Leading zeroes may be omitted when programming address T: T101=T0101

– If 0 is programmed for tool or only 1 or 2 digits are written at address T tool is not changed, only a new offset group is called. Meaning of T12: Use offset number 12.

When a move command and a tool number (T) are programmed in a given block, T function will be issued during or after execution of the move command. The execution method is determined by the builder of the machine.

# 12 Miscellaneous and Auxiliary Functions

## 12.1 Miscellaneous Functions (Codes M)

With a numerical value of max. 3 digits specified behind address **M**, the NC will transfer the code to the PLC.
When a movement command and a miscellaneous function (M) are programmed in a given block, function M will be issued during or after the motion command. The machine tool builder will define the way of execution.
Codes M's include standard functions, that can be used for special selected functions only. They:

**M00, M01, M02, M30, M96, M97, M98, M99**: Program control codes
**M03, M04, M05, M19**: Spindle rotation codes
**M07, M08, M09**: Coolant management codes
**M11, ..., M18**: Spindle-range changes codes

The rest of M values can be used without restrictions.
When indexing is triggered by M, the M codes of spindle indexing are selected on the basis of a parameter.
The control system enables several M codes of different groups to be written in a given block. In this case, however, codes M's have a fixed sequence of execution. The groups and the sequence of execution:

| | |
|---|---|
| group 1 | M11, ..., M18 (spindle gear range change) |
| group 2 | M03, M04, M05, M19 (spindle management) |
| group 3 | M07, M08, M09 (coolant management) |
| group 4 | M*nnn* (any other function M) |
| group 5 | codes M of spindle indexing |
| group 6 | M00, M01, M02, M30, M96, M97, M98, M99 (program control codes) |

The number of M functions that can be programmed in a given block is 5. Only one M of each group can be programmed in a block. Conflicting programming will produce error message *3032 CONFLICTING M CODES*.
The exact functioning of each M code is defined by the builder of the particular machine-tool to meet its specific requirements. The only exceptions are the program control codes.

The program control M codes are:

**M00**= programmed stop

The stop condition will be generated at the end of the block, in which M00 has been specified. All modal functions remain unchanged. It can be restarted by START.

**M01**= conditional stop

Its effect is identical with that of code M00. It will be executed when the **CONDITIONAL STOP** key is activated. Unless the appropriate key is set up, it is ineffective.

**M02, M30**= end of program

It means the end of the main program. All operations are stopped, and the control is "reset". The machine will be reset by the PLC program. Unless the parameter *PRTCNTM* differ from it, each of executed M02 or M03 command increase the counters of workpiece.

**M98**= call of a subprogram (subroutine)

It will call a subprogram (subroutine).

**M99**= end of subprogram (subroutine)

It will cause the execution to return to the position of call.

## 12.2 Auxiliary Function (Codes A, B, C)

Max. three digits can be specified at each of addresses **A, B, C** provided one (or all) of those addresses is (are) selected as auxiliary function(s) in parameters. The value specified for the auxiliary function will be transferred to the PLC.
When a movement command and an auxiliary function are programmed in a given block, function A, B, C will be issued during or after the motion command. The machine tool builder will define the way of execution.
For example, an indexing table can be indexed at address B.

## 12.3 Sequence of Execution of Various Functions

The various functions written in a given block will be executed by the control in the following sequence:

| | | |
|---|---|---|
| 1. | Tool call: | T |
| 2. | Spindle range selection: | M11, ..., M18 |
| 3. | Spindle speed: | S |
| 4. | Spindle management: | M03, M04, M05, M19 |
| 5. | Coolant: | M07, M08, M09 |
| 6. | Other function M: | M*nnn* |
| 7. | Spindle indexing: | With function M |
| 8. | Function A: | A |
| 9. | Function B: | B |
| 10. | Function C: | C |
| 11. | Program control codes: | M00, M01, M02, M30, M96, M97, M98, M99 |

If the above sequence of executions is not desirable, the block has to be broken up into several ones, with the functions written in the desired sequence in each block.

# 13 Part Program Configuration

The structure of the part program has been described already in the introduction presenting the codes and formats of the programs in the memory. This Section will discuss the procedures of organizing the part programs.

## 13.1 Sequence Number (Address N)

The blocks of the program can be specified with serial or sequence numbers. The numbering can be accomplished at address **N**. The blocks can be numbered with max. 5 digits at address N. The use of address N is not mandatory. Some blocks can be numbered, others not. The block numbers need not follow each other in a consecutive order.

## 13.2 Conditional Block Skip

A conditional block skip can be programmed at the slash address **/**. The value of the slash address may be 1 to 9. Digit 1 to 9 represents serial number of switches .
Switch **CONDITIONAL BLOCK** No. 1 can be found on the operator's panel of control.
The other switches may be provided optionally, their signals can be entered through the interface of the control system.
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 switch conditional block skip is intended to be effective even in the block preceding the conditional block reset parameter 1248 CNDBKBUF to 0. Then commands conditional block skip (blocks beginning with character /) **suppress block buffering**. In this case **path will be distorted** when using it in state **G41**, **G42** but switch can be turned on even **during the execution of the previous block**.

If block buffering is intented that is command / not to suppress buffering set parameter 1248 CNDBKBUF to 1. Then commands conditional block skip (blocks beginning with character /) **do not suppress** block buffering. In this case **path will not be distorted** when using it in state **G41**, **G42** but switch should be turned on **before sarting the program** to be safely effective.

## 13.3 Main Program and Subprogram

Two different programs are differentiated - main program and subprogram. Repetitive activities may be involved in machining a component part, that can be described with a particular program detail. In order to avoid writing the repetitive program detail over and over again in the program, they can be organized into a subprogram to be called from the part program. The structures of the main program and the subprogram have been described in the Introduction.
The difference between them is that, whereas the machining is completed after execution of the main program the control is awaiting another START, while after execution of the subprogram it will return to the calling program, resuming the machining process at that point.
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 specified with codes M02 or M30 (not mandatory ones), whereas the subprogram must be terminated with code M99.

### 13.3.1 Calling the Subprogram

The series of instructions

**M98 P....**

will generate a subprogram call. As a result, the execution of the program will be resumed at the subprogram, the number of which is defined at address P. The limit of address P are 1 to 9999. After the execution of the subprogram machining will be continued in the main program with the block following the subprogram call.

```
main program                      subprogram        comment

O0010                                               execution of (main-)
......                                              program O0010
......
M98 P0011         --->            O0011             calling subprogram
                                                    O0011
                                  ......            execution    of
                                  ......            subprogram O0011
                                  ......
next block        <---            M99               return to the calling
                                                    program
......                                              resumption of program
......                                              O0010
```

The series of instructions

**M98 P.... L....**

will call the subprogram (specified at address P) repeteatedly in succession specified at address L. The limit of address L is 1 to 9999. Unless L is assigned a value, the subprogram will be called once, i.e., the control will assume L=1.

Instruction M98 P11 L6 means that subprogram 011 has to be called six times repeatedly.

It is also possible to call a subprogram from another subprogram. The subprogram calls can be "nested" to max. 4 levels.

```
main program       subprogram  subprogram  subprogram  subprogram
O0001          ┌──>O0011    ┌──>O0012   ┌──>O0013   ┌──>O0014
....           │   ....     │   ....    │   ....    │   ....
....           │   ....     │   ....    │   ....    │   ....
M98P11 ────────┘   M98P12 ──┘   M98P13 ─┘   M98P14 ─┘   ....
....<─────────┐    ....<───┐    ....<──┐    ....<───┐   ....
....          │    ....    │    ....   │    ....    │   ....
M02           └─── M99     └─── M99    └─── M99     └── M99
```

☞ *Notes*:

– An error message *3069 LEVEL EXCESS* is returned when the number of subprogram calls "nested" exceeds 4.

– An error message *3071 MISSING OR FAULTY P* is returned when the value of address P exceeds 9999 or is not specified.

– An error message *3072 DEFINITION ERROR L* is returned when the value of L is incorrect.

– An error message *3073 NOT EXISTING PROGRAM* is returned when a program specified with an identifier at address P is not available in the memory.

## 13.3.2 Return from a Subprogram

The use of instruction

**M99**

in a subprogram means the end of that subprogram, and the program execution returns to the block following the call in the calling program.

```
main program                       subprogram        comment

O0010                                                execution of program
......                                               O0010
......
......
N101 M98 P0011    --->             O0011             calling subprogram
                                                     O0011
                                   ......            execution of
                                   ......            subprogram O0011
                                   ......
N102 ......       <---             M99               return to the next
                                                     block of the calling
                                                     program
......                                               resumption of program
......                                               O0010
```

The use of instruction

**M99 P...**

in a subprogram means the end of that subprogram, and the program execution returns to the block of calling program specified at address P. In this case the limit values of P are to 99999.

```
main program                       subprogram        comment

O0010                                                execution of program
......                                               O0010
......
......
N101 M98 P0011    --->             O0011             calling subprogram
                                                     O0011
                                   ......            execution of
                                   ......            subprogram O0011
                                   ......
N250 ......       <---             M99 P250          return to the N250
                                                     block of the calling
                                                     program resumption of
                                                     O0010
......
......
```

Instruction

**M99 (P.....) L....**

will rewrite the cycle counter of the calling program. With 0 written for L, the subprogram will be called only once. If, e.g., subprogram O0011 is called with instruction M98 P11 L20, and a return is made with instruction M99 L5, subprogram O0011 will be called 6 times. (The limit values of L are 1 to 9999.)

☞ *Note*:
– An error message *3070 NOT EXISTING BLOCK NO. P* is displayed when the return block number (P) is not found in the calling program.

### 13.3.3 Jump within the Main Program

The use of instruction

**M99**

in the main program will produce an unconditional jump to the first block of the main program, and the execution of the program will be resumed there. The use of this instruction results in an endless cycle:

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

The use of instruction

**M99 P.....**

will produce an unconditional jump to the block specified at address P of the main program, and the execution of the program will be resumed there. The use of this instruction may result in an endless cycle:

```
O0011                                O0011
....                                 ....
....                                 M99 P225
N128....<                            ....
....                                 ....
....                                 N225  <
M99 P128                             ....
```

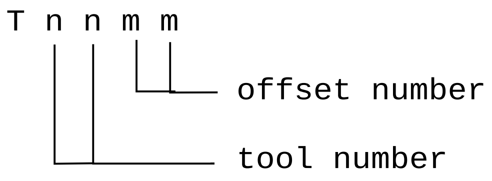
The possibility of endless cycles can be avoided by specifying the block containing instruction M99 in the form /1 M99. Now the jump will be omitted or not, depending on the setting of the conditional block skip switch.

# 14 The Tool Compensation

In order not to take the overhang values, tool radii ect. belonging to the different tools into account, the tool characteristics are gathered in a table, namely in the offset table. In case a tool is called in the part program, the place in the offset table the data of the given tool can be found must be given. Henceforth the control leads the tool in the programmed path by acknowledging the offsets referred to.

## 14.1 Reference to Tool Offset

Reference to tool offset can be made by the second two digits of the number written at address T.
Interpretation of the code written at address T:



Meaning of command T1236: Activate tool No. 12 and call group No. 36 of the offset table
– the leading zeroes may be left when programming address T: T101=T0101
– If 0 is programmed for the tool number or only one or two digits are written at address T, no tool replacement occurs, but a new offset group is used. The meaning of T12: Use offset group 12.

By the use of the offset number a group of the tool offset table is selected for the control. The elements of this table are as follows:

| Group number | X | | Y | | Z | | R | | Q |
|---|---|---|---|---|---|---|---|---|---|
| | geom. | wear | geom. | wear | geom. | wear | geom. | wear | |
| 1 | 123.500 | -0.234 | 87.450 | -0.129 | 267.400 | -0.036 | 1 | -0.010 | 3 |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| ... | | | | | | | | | |
| 47 | | | | | | | | | |

The tool offset table contains the components of the tool vector in X and Z directions (optionally also in Y direction), the tool nose radius (R) as well as the tool state code (Q).

☞ *Warning*

*Offset group 00 is not in the table, its offset values are always zeroes, thus offset group 00 deletes the offset vector.*

Offsets in X, (Y), Z directions and radius compensations (R) can be two types: Geometry and wear offsets.

**Geometry value**: Length/radius of the measured tool, signed number.

**Wear value**: Amount of wear occurring in the course of machining, signed number.

If an offset group is referred to in program at address T the control always takes the sum of geometry and wear values as offset amount into account.

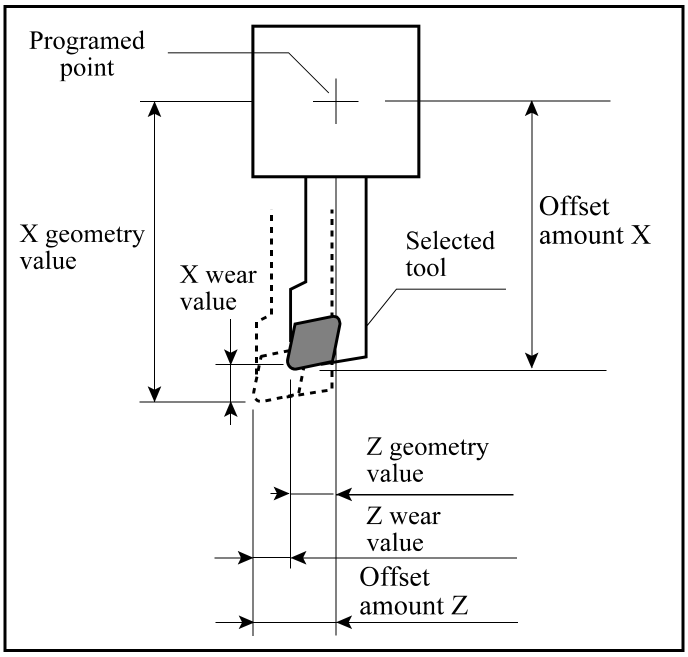


Fig. **14.1**-1

For example if reference to Tnn01 is made in the program the offset value for X axis on the basis of the above table according to line 01: 123.500+(–0.234)=123.266

The same goes for offset values in Z direction (and Y) and also for tool radius offset values (R).

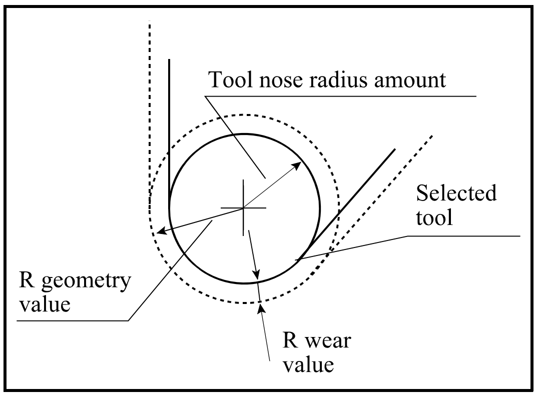


Fig. **14.1** -2

Value limits of geometric and wear offset values:

| input unit system | output unit system | increment system | geometry value | wear value | di-mension |
|---|---|---|---|---|---|
| mm | mm | IR-A | ±0.01 ÷99999.99 | ±0.01÷163.80 | mm |
| | | IR-B | ±0.001÷9999.999 | ±0.001÷16.380 | |
| | | IR-C | ±0.0001÷999.9999 | ±0.0001÷1.6380 | |
| inch | mm | IR-A | ±0.001÷9999.999 | ±0.001÷6.448 | inch |
| | | IR-B | ±0.0001÷999.9999 | ±0.0001÷0.6448 | |
| | | IR-C | ±0.00001÷99.99999 | ±0.00001÷0.06448 | |
| inch | inch | IR-A | ±0.001÷9999.999 | ±0.001÷16.380 | inch |
| | | IR-B | ±0.0001÷999.9999 | ±0.0001÷1.6380 | |
| | | IR-C | ±0.00001÷99.99999 | ±0.00001÷0.16380 | |
| mm | inch | IR-A | ±0.01÷99999.99 | ±0.01÷416.05 | mm |
| | | IR-B | ±0.001÷9999.999 | ±0.001÷41.605 | |
| | | IR-C | ±0.0001÷999.9999 | ±0.0001÷4.1605 | |

The direction of imaginary tool nose viewed from tool nose center must be also given in the offset table. Length compensations X and Z are measured to the imaginary tool nose.

The imaginary tool nose number (Q) is a one-digit number, its value may be 0,1,...9.

The imaginary tool nose number depends on the state of the applied coordinate system. The below two figures show the imaginary tool nose numbers (Q) to be used.

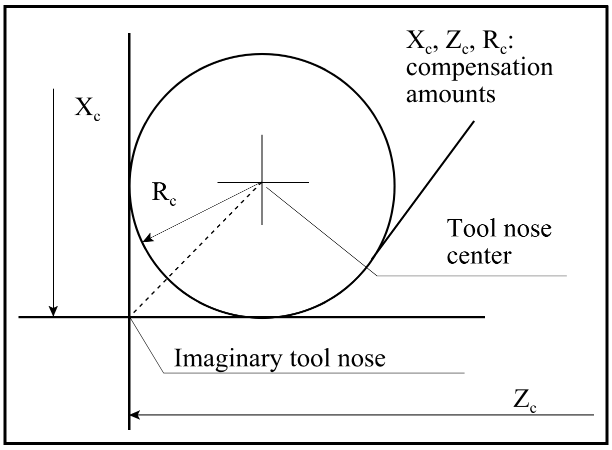


Fig. **14.1-3**

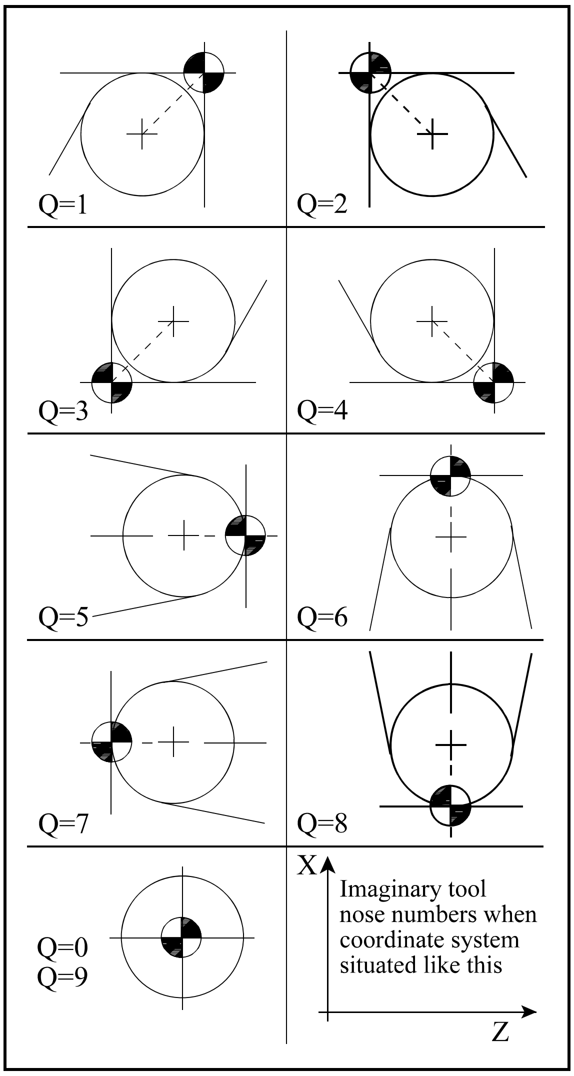

Q=1
Q=2
Q=3
Q=4
Q=5
Q=6
Q=7
Q=8
Q=0
Q=9
X
Imaginary tool nose numbers when coordinate system situated like this
Z

Fig. **14.1-4**

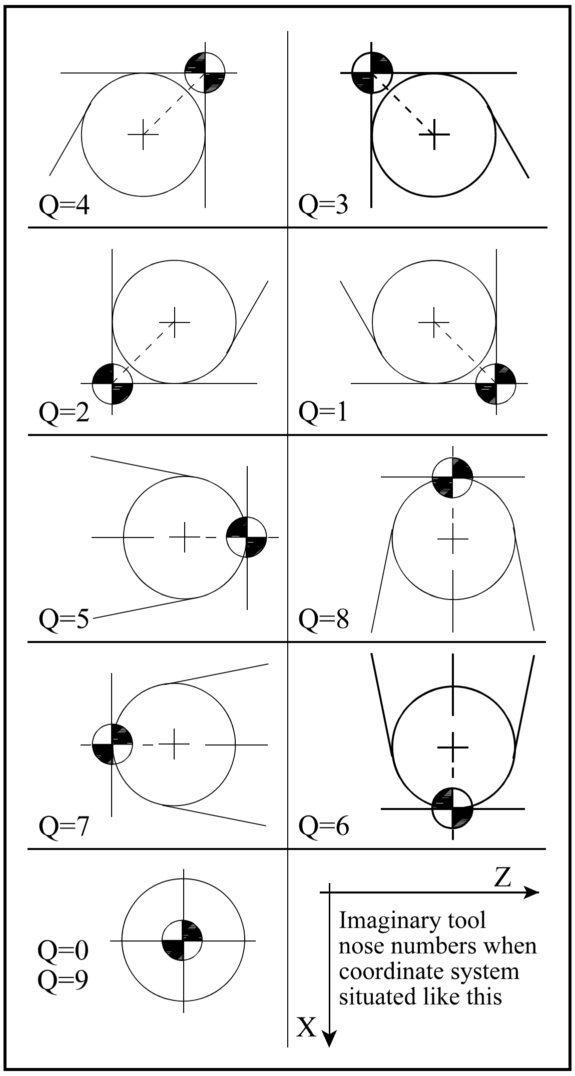

Q=4
Q=3
Q=2
Q=1
Q=5
Q=8
Q=7
Q=6
Q=0
Q=9
Z
Imaginary tool nose numbers when coordinate system situated like this
X

Fig. **14.1-5**

The compensation code called is modal, thus the control takes the same offset amounts into account until an other T command is received, i.e. when the compensation values are read by means of a T command, in this case the modification of the offset table (for example by programming G10) does no longer have an effect on the value read.
The offset values of the tool compensation memory are backed up upon power-off.
The tool offset memory can also be saved as a part program in the memory.
The tool offset values can be set, as well as modified from control panel by means of data input and from program by the use of setting command G10. If the current offset value is modified with the help of command G10 the current offset register must be re-referred to, because the modified value is taken into account only in this case.

## 14.2 Modification of Tool Offset Values from the Program (G10)

Block

**G10** L P X Y Z R Q
**G10** L P XI YI ZI RI Q or
**G10** L P U V W C Q

can be used for modifying the tool offset values from the program. G10 is a single-shot command. The addresses and their values have the following meanings:

**L=10**: Entering geometry value
**L=11**: Entering wear value

The number of offset group is specified at address **P**.
In case of absolute data definition (state G90) the length compensation values and radius compensation values are specified at addresses **X, Y, Z, R**. If the compensation value is to be modified in increment G91 must be programed or increment operator **I** must be used after the appropriate address or instead of addresses X, Y, Z, R addresses **U, V, W, C** must be applied. In case of incremental programming the given value is added with sign to the current offset value.
The imaginary tool nose number is given at address **Q** (Q=0...9).

## 14.3 Taking the Tool Length Offset into Account

As the effect of T code the coordinate system is shifted by values $X_c$, $Y_c$, $Z_c$ belonging to the offset group specified in the T code. This means that the offset value in $X_c$, $Y_c$, $Z_c$ direction belonging to the selected offset group is subtracted from the current position X, Y, Z. Henceforth it is not the reference point of the tool holder but the coordinates of the imaginary tool nose displayed on the screen. From now on offset value is always referred to as the sum of geometry and wear offsets.

For example as the effect of command lines

```
(T0000)
N10 G0 (G90) X700 Z350
N20 T202
N30 X300 Z150
```

the reference point of the tool holder is led to the position of X700; Z350 coordinates in block N10. The tool is activated and offset values $X_c$=340, $Z_c$=30 belonging to compensation group No. 2 are subtracted from the position in block N20. In this case the position display changes to value X=700–340=360; Z=350–30=320. *No movement is executed in block N20.* In block N30 it is already the imaginary tool nose led to programmed position X300; Z150, i.e. the movement done is X=300–360= –60; Z=150–320= –170.

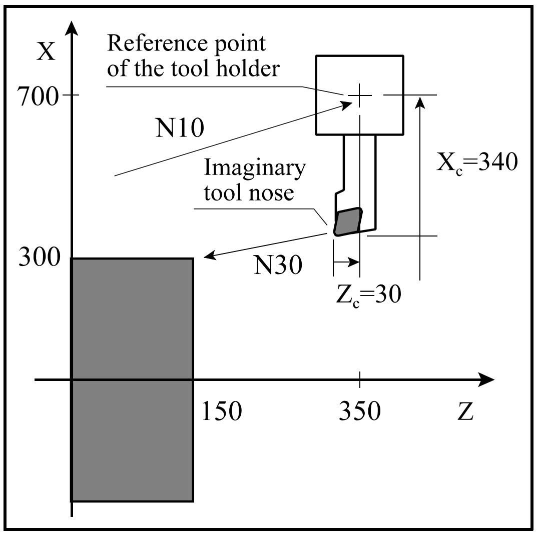


Fig. **14.3-1**

If T code is defined together with a move command the block end point is positioned by taking the new offset value determined by the T code into consideration. However the tool replacement is executed parallel to or at the end of the movement, provided the T code also contains tool replacement command. (The tool replacement timing is determined by the machine tool builder.)

Let us see the above example:
```
        (T0000)
        N10 G0 (G90) X700 Z350
        N20 X300 Z150 T202
```
In this case block N30 is left and the commands of blocks N20 and N30 are pooled. In block N20 in the course of movement it is already the imaginary tool nose led to the position of X=300, Z=150 coordinates as in the case of block N30 in the previous example. However tool replacement occurs either during or at the end of movement. The latter case may however easily lead to collision, if, e.g., the overhang of the previous tool is longer than that of the new one and tool replacement is executed by the machine at the end of block. For the above reason it is only expedient to program T code together with movement for calling new offset:
```
        N10 G0 (G90) X700 Z350
        N20 X300 Z150 T02
```
In the above example no tool replacement is implemented, only offset group No. 2 is called in block N20.

☞ *Warning!*

*As a consequence of the above, in case the T code results in tool replacement, e.g. turret rotation it is expedient to program the T code in a single block. If the T command executes only offset group change, T code may safely be programmed also in move command.*

Canceling of tool length compensation

Offset group No. 0 has a special role, it implements compensation canceling.
As the effect of command

**Tnn00** or
**T0**

the length compensation is canceled (nn: optional tool). The process is the reverse of that of compensation call. This means that offset values $X_c$, $Y_c$, $Z_c$ valid previously are added to X, Y, Z imaginary tool nose coordinates and hereafter the reference point coordinates of tool holder are seen in the display. If compensation canceling occurs together with move command the reference point of tool holder is sent to the programmed block end position.

For example in sample example
```
        N10 X180 Z120 T202
        N20 X200 Z180
        N30 X280 Z210 T200
```
the movement is compensated by the offset group called as compared to the programmed one in block N10, while the offset, i.e. the one set in block N10 is canceled in block N30.

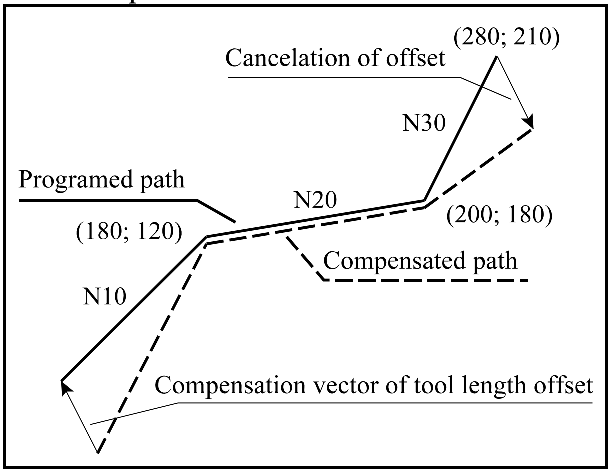


Fig. 14.3-2

## 14.4 Tool Nose Radius Compensation (G38, G39, G40, G41, G42)

If only tool length compensation is used, it is not possible to turn an accurate tapered line or a circular arc. In this case the imaginary tool nose is guided by the control along the programmed path. Because each tool's nose has a rounding, the part will be accurate where the path is parallel to one of the axes X or Z. Processing tapered lines or arcs insufficient depth of cut is turned, as it can be seen on the figure beside.

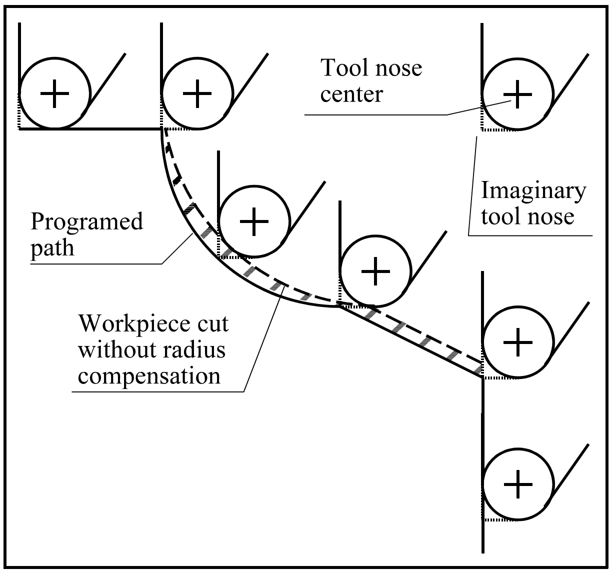


Fig. **14.4-1**

To be able to turn the contour of a workpiece and to specify the points of that formation as per the drawing in the program (regardless of the radius of the tool nose employed), the control must guide the tool nose center parallel to the programmed contour, spaced by the tool nose radius from the latter. The control will determine the distance between the path of the tool nose center and the programmed contour in accordance with the compensation value of the tool nose radius referred to by the compensation number of address T.

Before turning on the tool nose radius compensation the imaginary tool nose position is registered by the control. The imaginary tool nose number given at address Q in compensation table is needed to specify the direction of tool nose circle center from the imaginary nose. The tool radius (R) is added to or subtracted from the imaginary tool nose position in X and Z directions on the basis of imaginary tool nose number (position A in the figure). Afterwards the tool nose circle center is led to the end point of the compensation vector whose length is R and positioned perpendicular to the start point of the programed path (position B in the figure) Next the tool nose circle center is led parallel to the programmed path of a distance of R from it.

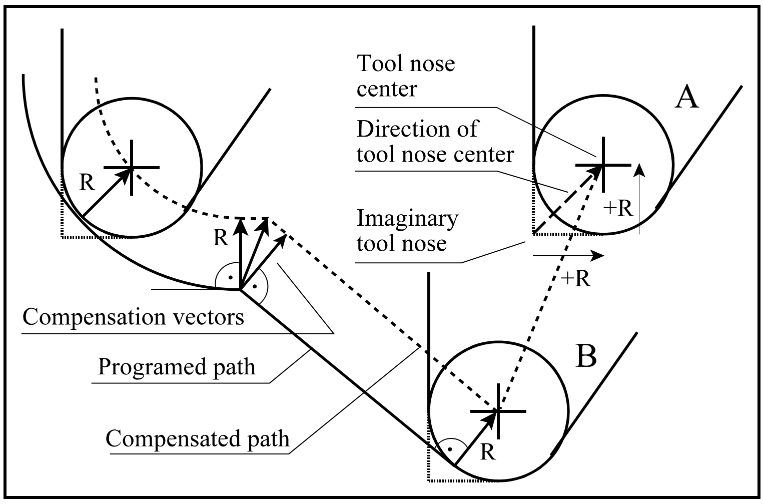


Fig. **14.4-2**

The compensation vector is a two-dimensional vector computed over and over again by the control in each block, modifying the programmed displacements with the compensation vectors effective at the beginning and end of each block. The length and direction of each compensation vector obtained vary with the compensation value (called at address T) and the geometry of the

transition between the two blocks.
The compensation vectors are computed in the plane selected by instructions G17, G18, G19. This is the plane of tool nose radius compensation. Movements outside of this plane are not influenced by compensation. If, e.g., plane X, Z is selected in state G18, the compensation vectors will be computed in that plane. In this case any movement in Y direction it will be unaffected by the compensation.
The compensation plane may not be changed while a tool radius compensation is being computed. Any attempt to do so will result in an error message *3010 PLANE SELECT. IN G41, G42* by the control.
If a compensation plane is to be defined with additional axes, they have to be defined as parallel ones in parameters. If, e.g., U is assumed as a parallel axis and the tool radius compensation is to be applied in plane Z, U, that plane can be selected by the specification of G18 U_Z_.

**G40**: Tool nose radius compensation cancel
**G41**: Tool nose radius compensation left
**G42**: Tool nose radius compensation right

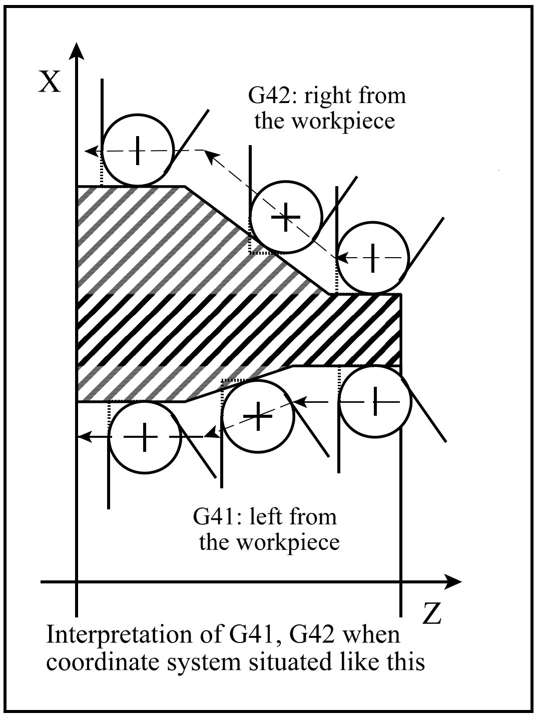

X
G42: right from the workpiece
G41: left from the workpiece
Z
Interpretation of G41, G42 when coordinate system situated like this

Fig. **14.4-3**

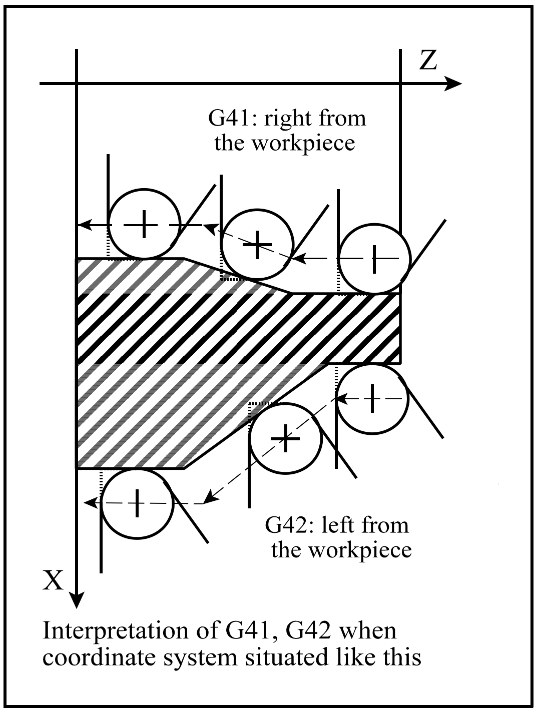

Z
G41: right from the workpiece
G42: left from the workpiece
X
Interpretation of G41, G42 when coordinate system situated like this

Fig. **14.4-4**

Command G41 or G42 will set up the compensation computation. In state G41 or G42 the pro grammed contours will be tracked from left side or right side (seen from the travel direction), respectively. The compensation number of tool radii has to be specified at the lower two digits of address T. The specification of Tnn00 or T0 is always equivalent to calling zero radius value. The compensation calculations are performed for interpolation movements G00, G01, G02, G03.

The above points refer to the specification of positive tool radius compensation, but its value may be negative, too. It has a practical meaning if, e.g., a given subprogram is to be used for defining the contours of a "female" part and of a "male" one being matched to the former. A possible way of doing this is to turn the female part with G41 and the male part with G42. However, that change-over may be omitted from the program when the female part is machined with a positive radius compensation, and the male part with a negative one. Now the path of the tool center is reversed with respect to the programmed G41 or G42.

| | Radius compensation: **Positive** | Radius compensation: **Negative** |
|---|---|---|
| **G41** | on the left side | on the right side |
| **G42** | on the right side | on the left side |

☞ *Note*:
– For simplicity's sake, the subsequent descriptions and Figures will always refer to positive radius compensations.

Command G40, Tnn00 or T0 will cancel the offset compensation. The difference between the two commands is that T0 deletes only the compensation vector, leaving state G41 or G42 unchanged. If a reference is made subsequently to an address Tnnmm (mm≠0), the compensation vector will be computed with the new tool radius as the function of state G41 or G42.
If, however, instruction G40 is used, the compensation vectors will not be calculated any more. The procedure of setting up and canceling the radius compensation is detailed in the subsequent sections.
Commands G40, G41, G42 are modal ones. The control will assume state G40 after power-on, at the end of a program or in the event of resetting the program to its beginning, under such conditions the radius compensation vectors will be deleted.
Radius compensation instructions will be carried out by the control in automatic mode only. It is ineffective when programming a single block in manual mode. The reason of this is as follows. For the control to be able to compute the compensation vector in the end point of a block (interpolation), it must also read the next block containing the movement in the selected plane. The compensation vector depends on the transition between the two interpolations. Accordingly, several blocks (interpolations) have to be pre-processed for the calculation of a compensation vector.

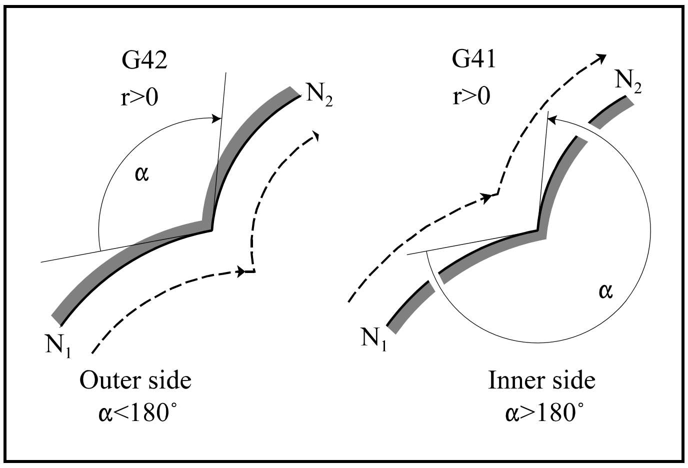


Fig. **14.4-5**

An auxiliary data is to be introduced before embarking on the discussion of the details of the compensation computation. It is "α", the angle at the corner of two consecutive blocks viewing from the workpiece side. The direction of α depends on whether the tool goes around the corner from the left or right side.

The control will select the strategy of going around in the intersection points as the function of angle α. If α>180°, i.e., the tool is working inside, the control will compute a point of intersection between the two interpolations. If α<180°, i.e., the tool is moving around the outside, it *may add* further straight sections.

### 14.4.1 Start up of Tool Nose Radius Compensation

After power-on, end of program or resetting to the beginning of the program, the control will assume state G40. The offset vector will be deleted, the path of the imaginary tool nose will coincide with the programmed path.
Under instruction G41 or G42 the control will exit from state G40 to enter in radius-compensation computation mode. State G41 or G42 will only be assumed in a block containing a linear interpolation (G00 or G01). The control will return error message *3043 G41, G42 IN G2, G3* to any attempt to set up the compensation calculation in a circular interpolation (G02, G03). The control will only choose the procedure of  the start up of tool nose radius compensation, if G41 or G42 was commanded after G40. In other words, the control will not adopt the start up procedure when the compensation is deleted with T00 and re-activated with Tnn (nn being a number other than 0).
The basic instances of starting compensation up depending on the angle of α at the corner of the two consecutive blocks and the type of interpolations (linear-to-linear, linear-to-circular) as shown below. The Figures refer to instance G42, positive radius compensation assumed.
☞ *Note*: The symbols in the Figures (below and afterwards) have the following meanings:
r: Value of radius compensation,
L: Straight line
C: Circular arc,
S: Single block stop point,
Dashed line: The path of tool nose center,
Continuos line: The programmed path.

<u>Basic instances of  starting up the tool nose radius compensation:</u>

| | |
|---|---|
| (G40) | (G40) |
| G42 G01 X_ Z_ | G42 G01 X_ Z_ |
| X_ Z_ | G2 X_ Z_ R_ |

Going around an inside corner, 180°<α<360°

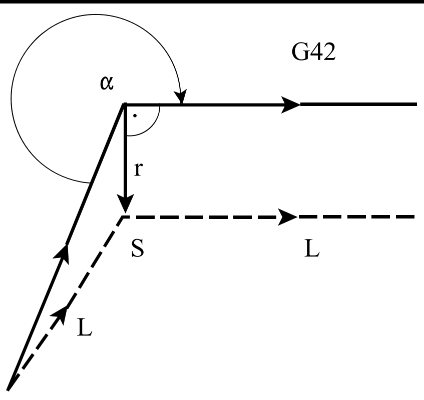
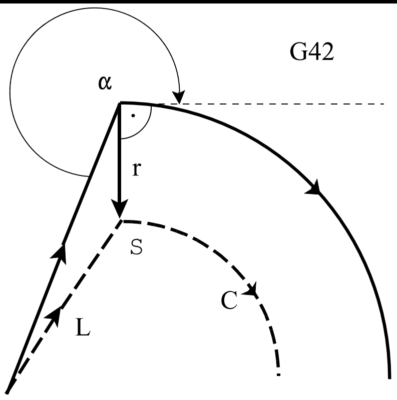
| Linear to linear | Linear to circular |
|---|---|
|  |  |

Fig. **<u>14.4.1</u>-1**

Going around the outside of a corner at an obtuse angle, 90°≤α≤180°

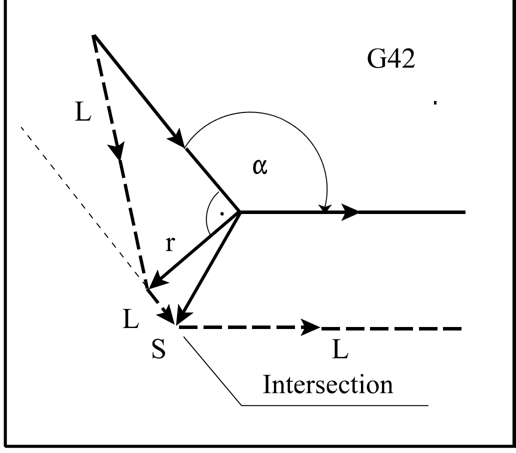
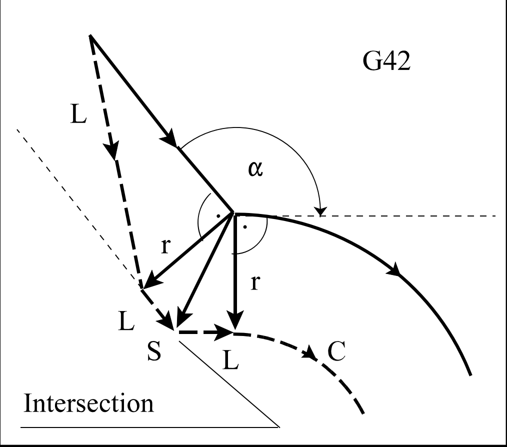
| Linear to linear | Linear to circular |
|---|---|
|  |  |

Fig. **14.4.1-2**

Going around the outside of a corner at an acute angle, 0°≤α<90°

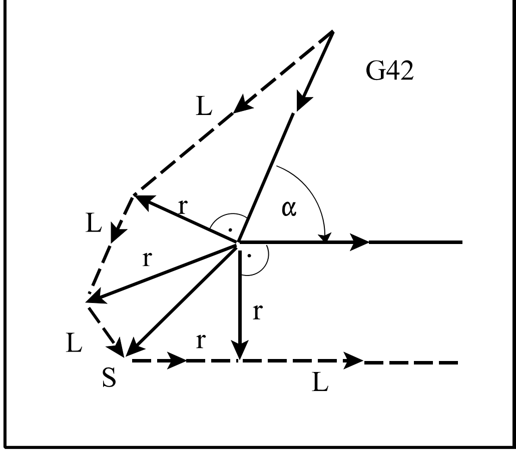
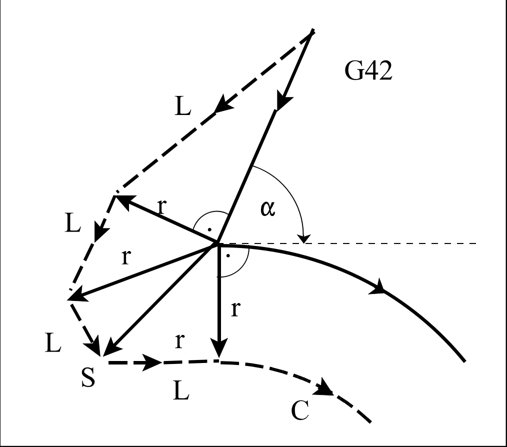
| Linear to linear | Linear to circular |
|---|---|
|  |  |

Fig. **14.4.1-3**

Special instances of starting up the radius compensation:

If values are assigned to I, J, K in the compensation-selecting block (G41 or G42) - but only to those in the selected plane (e.g., to I, K in the case of G18) - the control will move to the intersection point between the next block and the straight line defined by I, J, K with starting up radius compensation. The values of I, J, K are always incremental ones, the vector defined by them pointing to the end point of the interpolation, in which it has been programmed. This facility is useful, e.g., in moving to an inside corner.

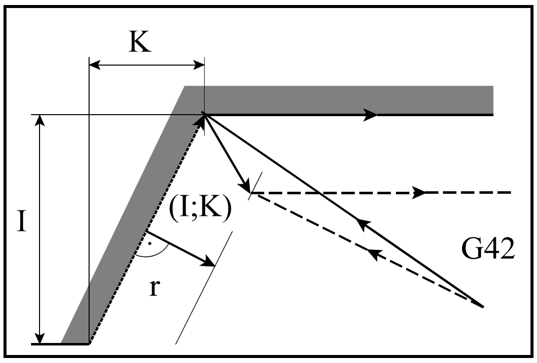


Fig. **14.4.1-4**

. . .

```
G91 G18 G40
...
N110 G42 G1 X120 Z–80 I70 K50
N120 Z100
...
```

In this case the control will always compute a point of intersection regardless of whether an inside or an outside corner is to be machined.

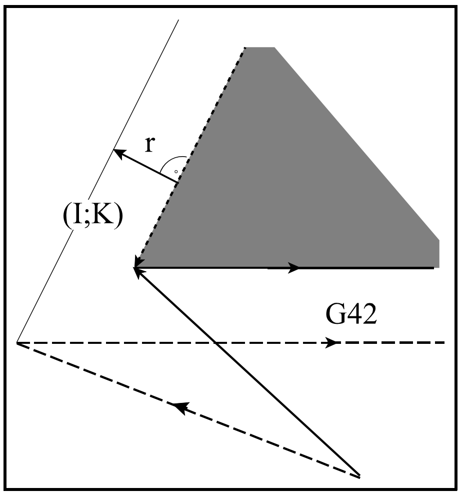


Fig. **14.4.1-5**

Unless a point of intersection is found, the control will move, at right angles, to the start point of the next interpolation.

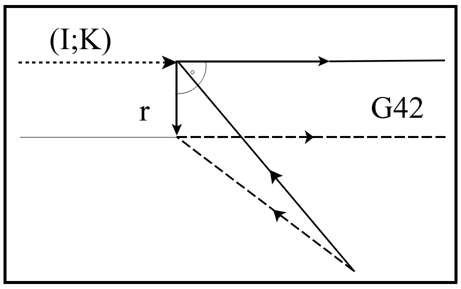


Fig. **14.4.1-6**

When the compensation is set up by a special block in which no movement is programmed in the selected plane, the compensation will be set up without any movement, the calculated compensation vector’s length is 0. The compensation vector is computed at the end of the next motion block according to the strategy corresponding to compensation computation in offset mode (see the next chapter).

```
      ...
      N10 G40 G18 G0 X0 Z0
      N15 G42
      N20 G1 Z80
      N25 X120 Z110
      ...
```

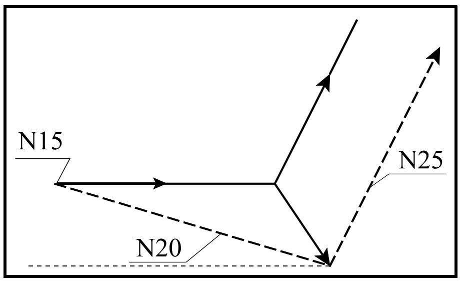


Fig. **14.4.1-7**

If zero displacement is programmed (or such is produced) in the block containing the activation of compensation (G41, G42), the control will not perform any movement but will carry on the machining along the above-mentioned strategy.

```
...
N10 G40 G18 G0 X0 Z0
N15 G91 G42 Z0
N20 G1 Z80
N25 X120 Z30
...
```

If a displacement of 0 is obtained in the selected plane in the block following the start-up of compensation, the compensation vector will be set at right angles to the interpolation performing the setting-up. The path of the tool in the next interpolation will be not parallel to the programmed contour:

```
...
N10 G40 G18 G0 X0 Z0
N15 G91 G42 Z80
N20 G1 z0
N25 X120 Z30
N30 Z60
...
```

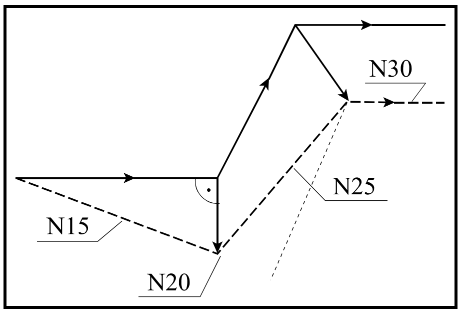


Fig. **14.4.1-8**

## 14.4.2 Rules of Tool Nose Radius Compensation in Offset Mode

In offset mode the compensation vectors will be calculated continuously between interpolation blocks G00, G01, G02, G03 (see the basic instances) until more than one block will be inserted, that do not contain displacements in the selected plane. This category includes a block containing dwell or functions.

Basic instances of offset mode:

Computation of intersection point for inside corners, 180°<α<360°

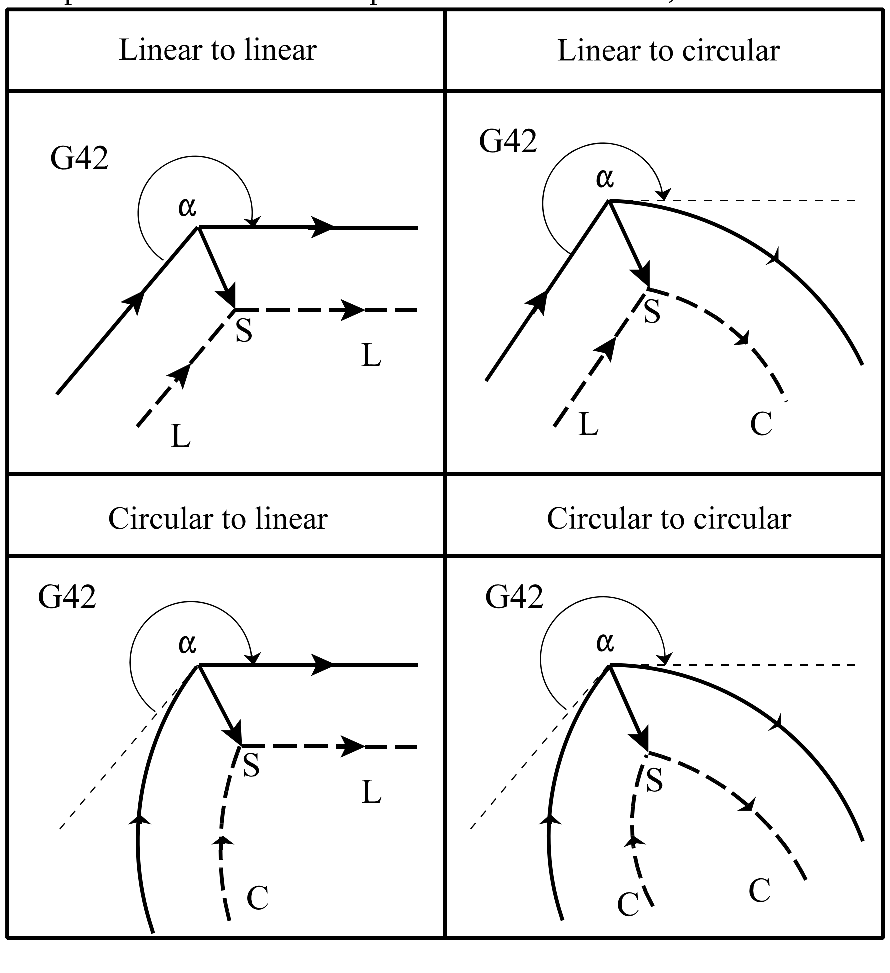


Fig. **14.4.2-1**

It may occur that no intersection point is obtained with some tool nose radius values. In this case the control comes to a halt during execution of the previous interpolation and returns error message *3046 NO INTERSECTION G41, G42*.

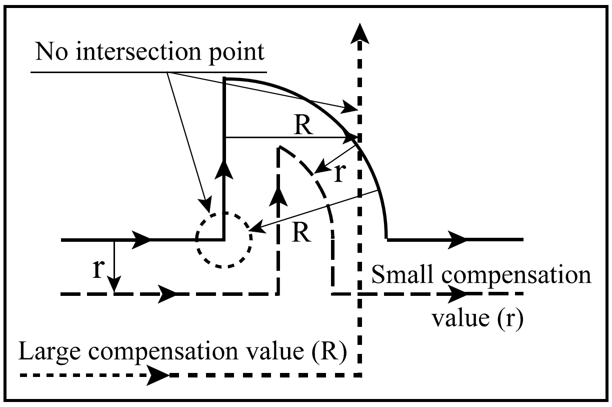


Fig. **14.4.2**-2

Going around the outside of a corner at an obtuse angle, 90°≤α≤180°

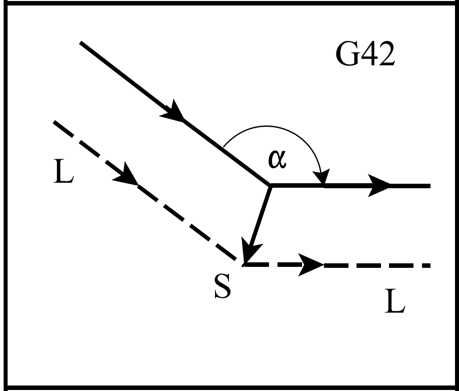
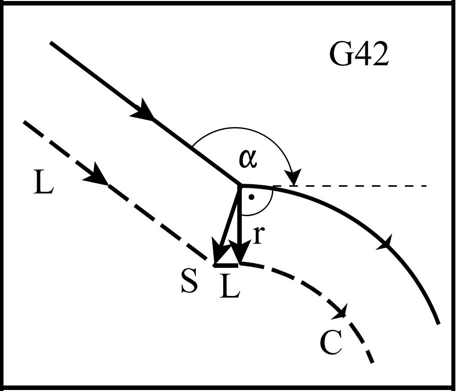
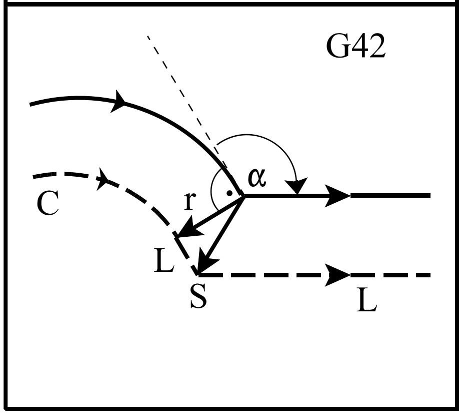
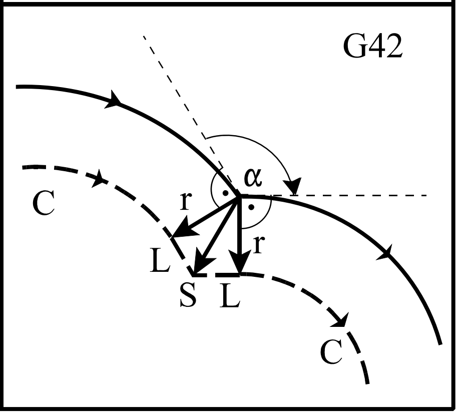
| Linear to linear | Linear to circular |
|---|---|
|  |  |
| Circular to linear | Circular to circular |
|  |  |

Fig. **14.4.2**-3

Going around the outside of a corner at an acute angle, 0°≤α<90°

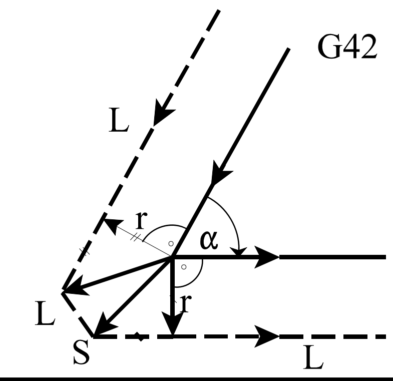
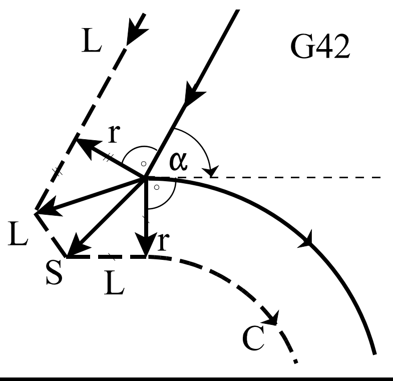
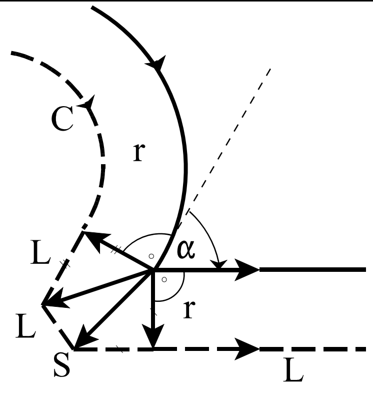
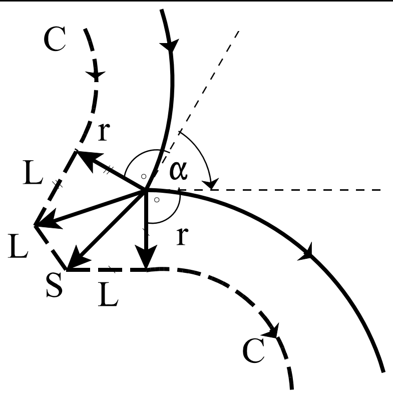
| Linear to linear | Linear to circular |
|---|---|
|  |  |
| **Circular to linear** | **Circular to circular** |
|  |  |

Fig. **14.4.2-4**

Special instances of offset mode:

If zero displacement is programmed (or such is obtained) in the selected plane in a block in offset mode, a perpendicular vector will be positioned to the end point of the previous interpolation, the length of the vector will be equal to the radius value. Instances of this kind should be handled with caution because of the hazards of inadvertent undercutting or distortions (in the case of a circle).

For example:

```
...G91 G18 G42...
N110 G1 X100 Z40
N120 Z0
N130 Z90
N140 X–40 Z50
...
```

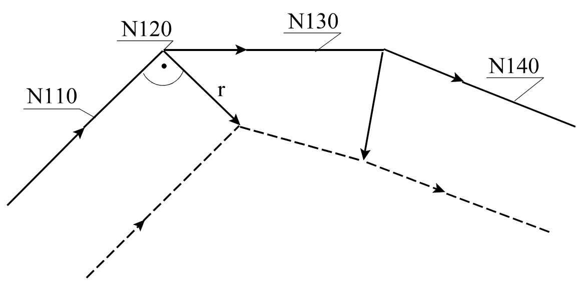


Fig. **14.4.2-5**

## 14.4.3 Canceling of Offset Mode

Command G40 will cancel the computation of tool radius compensation. Such a command can be issued with linear interpolation only. The control will return error message *3042 G40 IN G2, G3* to any attempt to program G40 in a circular interpolation.

Basic instances of canceling offset mode:

```
(G42)               (G42)
G01 X_ Z_           G02 X_ Z_ R_
G40 X_ Z_           G40 G1 X_ Z_
```

Going around an inside corner, 180°<α<360°

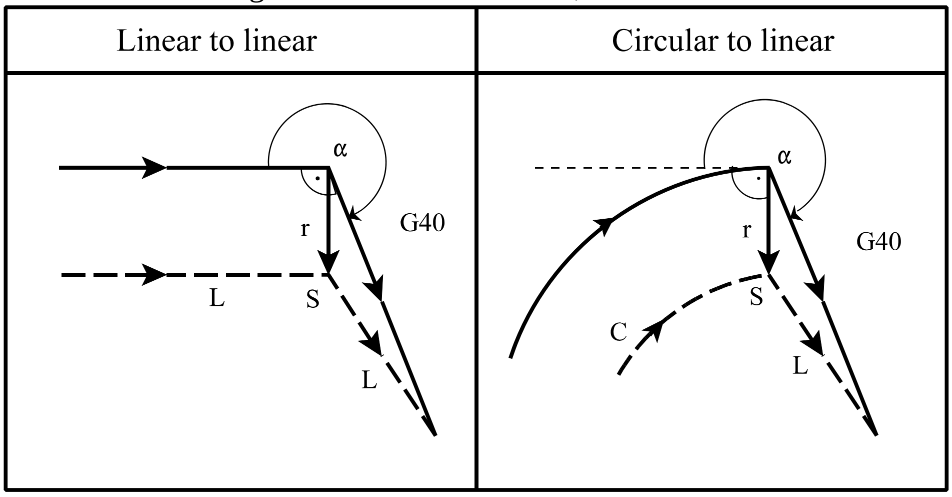


Fig. **14.4.3-1**

Going around the outside of a corner at an obtuse angle, 90°≤α≤180°

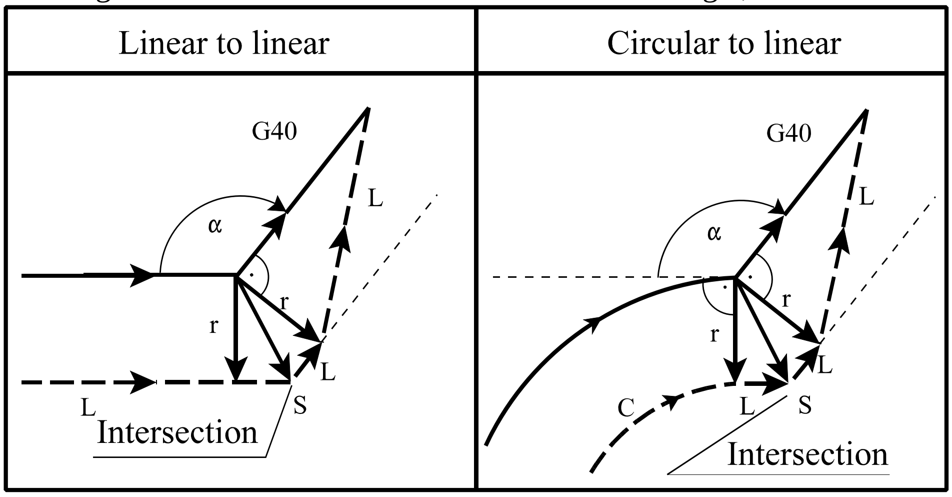


Fig. **14.4.3-2**

Going around the outside of a corner at an acute angle, $0° \leq \alpha < 90°$

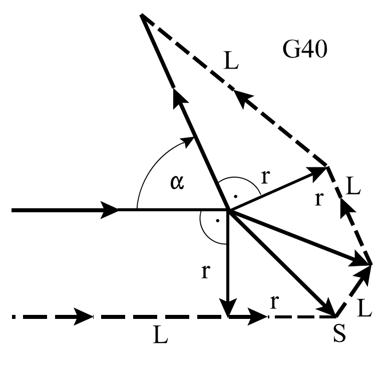
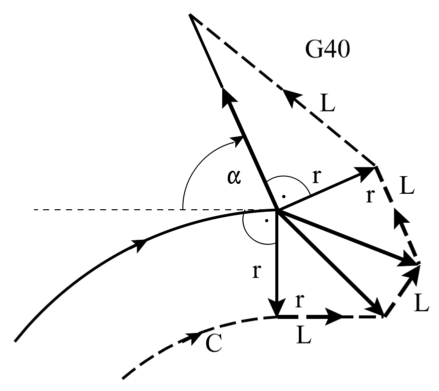
| Linear to linear | Circular to linear |
|---|---|
|  |  |

Fig. **14.4.3-3**

Special instances of canceling offset mode:

If values are assigned to I, J, K in the compensation cancel block (G40) - but only to those in the selected plane (e.g., to I, K in the case of G18) - the control will move to the intersection point between the previous interpolation and the straight line defined by I, J, K. The values of I, J, K are always incremental, the vector defined by them points away from the end point of the previous interpolation.

This facility is useful, e.g., for moving from an inside corner.

```
...
...G91 G18 G42...
N100 G1 X120 Z50
N110 G40 X–120 Z70 I–20 K100
...
```

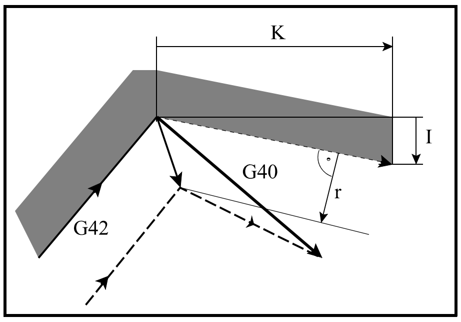


Fig. **14.4.3-4**

In this case the control will always compute a point of intersection regardless of whether an inside or an outside corner is to be machined.

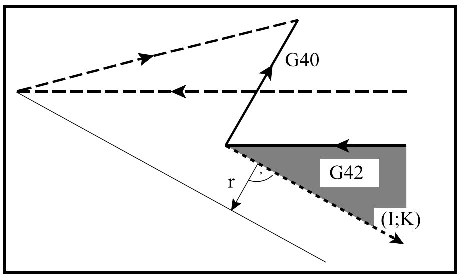


Fig. **14.4.3-5**

Unless a point of intersection is found, the control will move, at a right angle, to the end point of the previous interpolation.

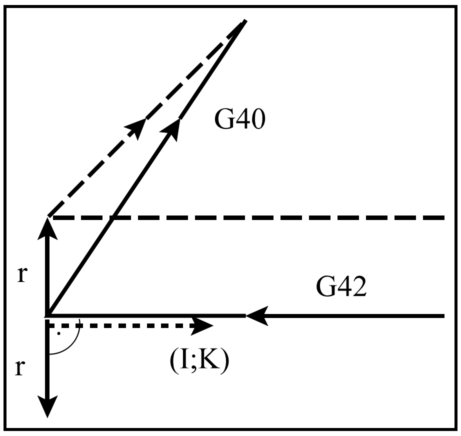


Fig. **14.4.3-6**

If the compensation is canceled in a block in which no movement is programmed in the selected plane, an offset vector perpendicular to the end point of the previous interpolation will be set and the compensation vector will be deleted by the end of the next movement block.

```
...G42 G18 G91...
N110 G1 X80 Z80
N120 G40
N130 X40 Z–70
...
```

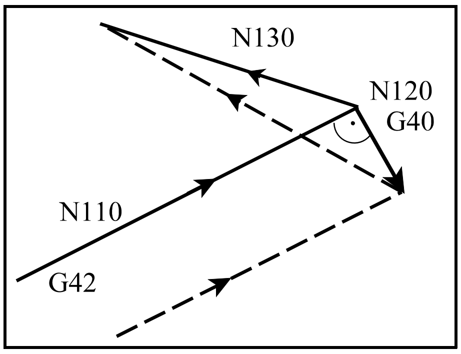


Fig. **14.4.3-7**

If zero displacement has been programmed (or such is obtained) in the block (G40) in which the compensation is canceled, an offset vector perpendicular to the end point of the previous interpolation will be calculated, and the control will cover that instance in block G40. For example:

```
...G42 G18 G91...
N110 G1 X80 Z80
N120 G40 X0
N130 X40 Z–70
...
```

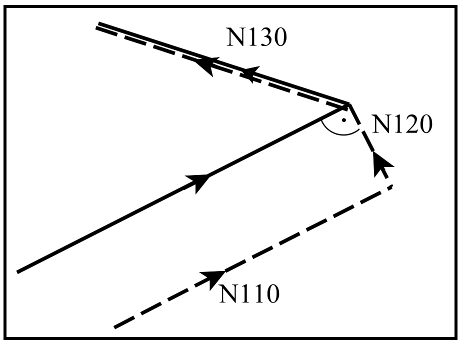


Fig. **14.4.3-8**

## 14.4.4 Change of Offset Direction While in the Offset Mode

The direction of tool-radius compensation computation is given in the Table below.

| | Radius compensation: **Positive** | Radius compensation: **Negative** |
|---|---|---|
| **G41** | left | right |
| **G42** | right | left |

The direction of offset mode can be reversed even during the computation of tool radius compensation. This can be accomplished by programming G41 or G42, or by calling a tool radius compensation of an opposite sign at address D. When the direction of offset mode is reversed, the control will not check it for being "outside", it will always calculate a point of intersection in first steps. The Figures below assume positive tool radii and change-overs from G42 to G41.

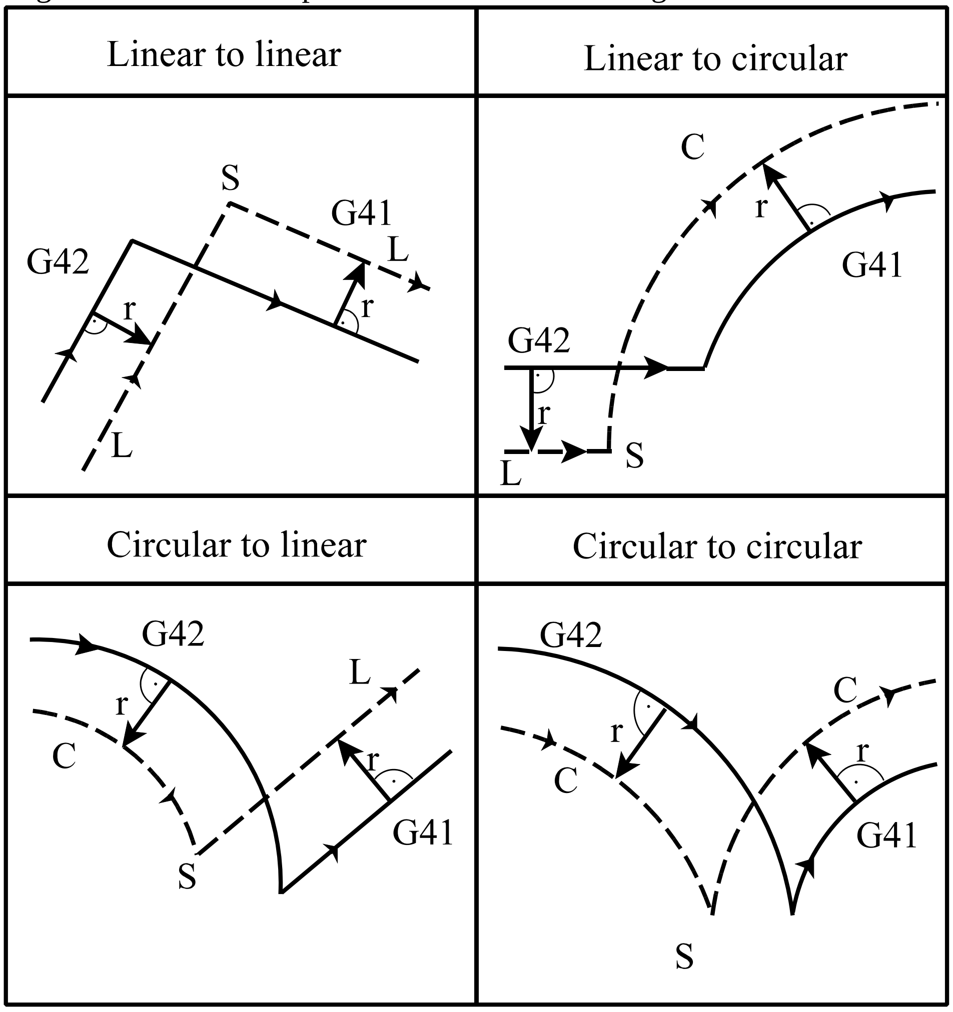


Fig. 14.4.4-1

Unless a point of intersection is found in a linear-to-linear transition, the path of the tool will be:

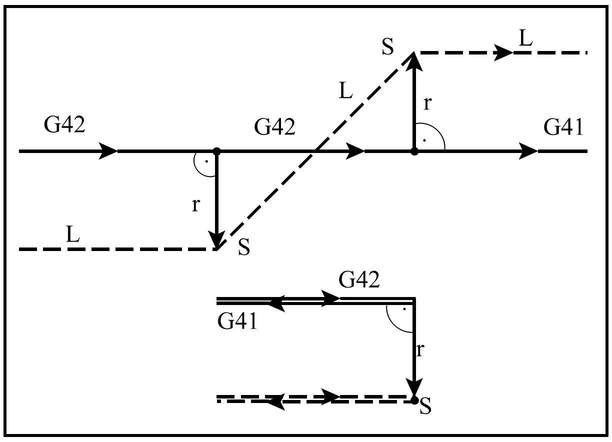


Fig. **14.4.4**-2

Unless a point of intersection is found in a linear-to-circular transition, the path of the tool will be:

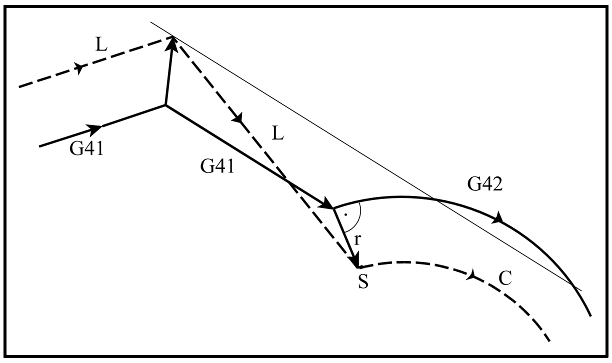


Fig. **14.4.4**-3

Unless a point of intersection is obtained in a circular-to-linear or circular-to-circular transition, the end of compensation vector in the start point of the first circular interpolation will be connected with the end point of the compensation vector perpendicular to the start point of the second interpolation by a circular arc of uncorrected programmed radius R. Now the center of the interconnecting circular arc will not coincide with the center of the programmed arc. The control will return error message *3047 CHANGE NOT POSSIBLE* if the direction change is not feasible even with the relocation of the circle center outlined above.

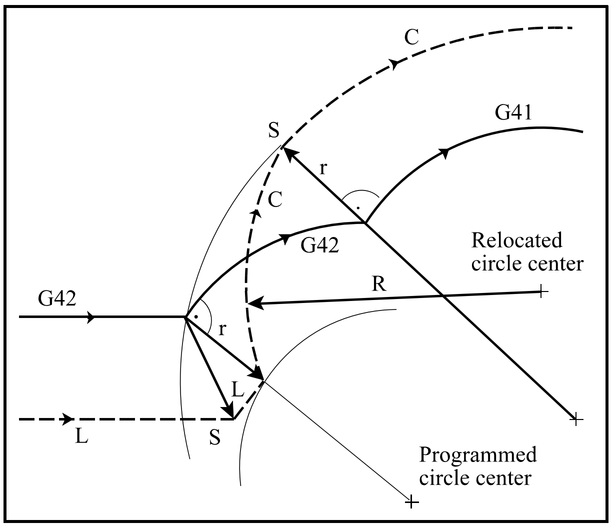


Fig. **14.4.4**-4

### 14.4.5 Programming Vector Hold (G38)

Under the action of command

**G38** v

the control will hold the last compensation vector between the previous interpolation and G38 block in offset mode, and will implement it at the end of G38 block irrespective of the transition between the G38 block and the next one. Code G38 is a single-shot one, i.e., it will not be modal. G38 has to be programmed over again if the vector is to be held in several consecutive blocks. G38 can be programmed in state G00 or G01 only, i.e., the vector-hold block must be invariably a linear interpolation, or else the control will return error message *3040 G38 NOT IN G0, G1*. Unless code G38 is used in offset mode (G41, G42), the control will return error message *3039 G38 CODE IN G40*.

An example of using G38:

```
...G18 G41 G91...
N110 G1 X120 Z60
N120 G38 X–80 Z90
N130 X140 Z20
N140 Z120
...
```

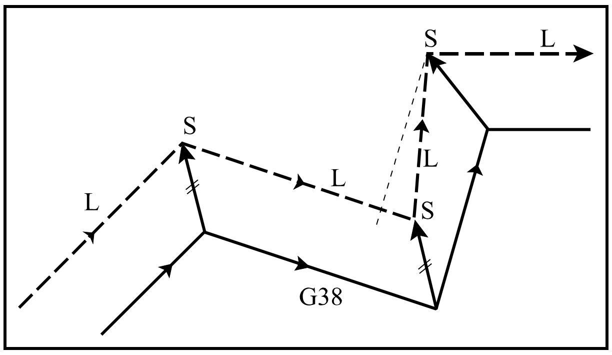


Fig. **14.4.5-1**

To program a recession without canceling the offset mode:

```
...G18 G42 G91...
N110 G1 Z40
N120 G38 Z50
N130 G38 X140
N140 G38 X-140
N150 Z60
...
```

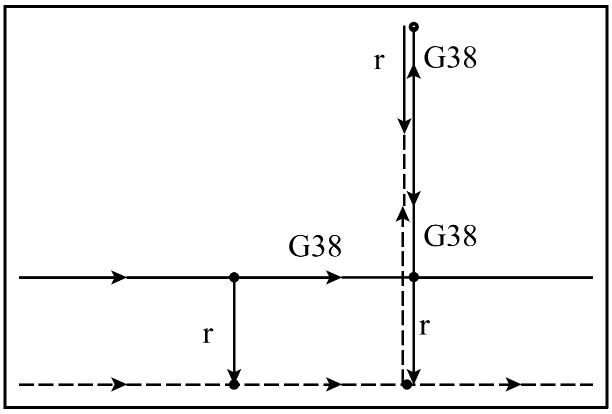


Fig. **14.4.5-2**

### 14.4.6 Programming Corner Arcs (G39)

By programming

**G39** (I J K),

it will be possible - in offset mode - to avoid the automatic intersection-point computation or the insertion of linear sections when going around outside corners, instead the tool center will travel along a circular arc equal to the tool radius.

It will insert an arc equal to the tool radius in direction G02 or G03 in state G41 or G42, respectively.

The start and end points of the arc will be given by a tool-radius long vector perpendicular to the end point of the path of previous interpolation and by a tool-radius vector perpendicular to the start point of the next one, respectively. G39 has to be programmed in a separate block:

```
      ...G18 G91 G41...
      N110 G1 Z100
      N120 G39
      N130 G3 X–160 Z80 K80
      ...
```

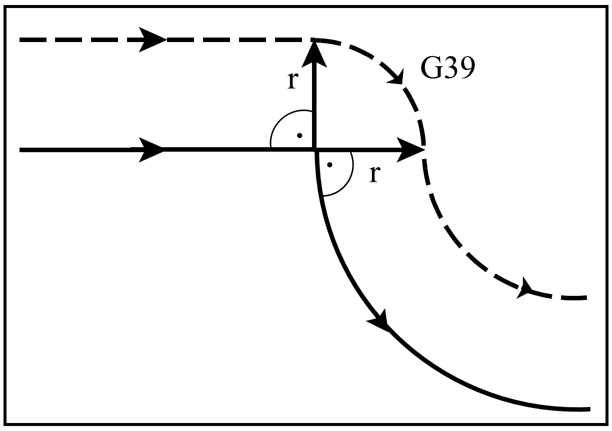


Fig. 14.4.6-1

When I, J or K is programmed in block G39, the end point of the circular arc will be given by a tool-radius long vector perpendicular to the vector defined by I, J or K from the end point of the previous interpolation, in accordance with the selected plane.

```
      ...G18 G91 G41...
      N110 G1 Z100
      N120 G39 I–60 K50
      N130 G40 X60 Z110
      ...
```

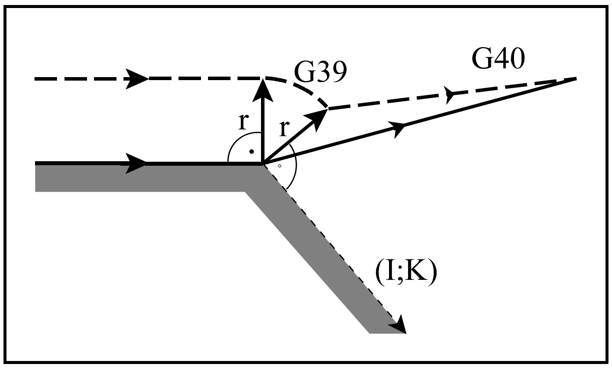


Fig. 14.4.6-2

The previously selected mirroring or rotating commands are effectual the vector defined by I, J or K. As a matter of fact, the scaling command will not affect the direction. No movement command can be programmed in a block of G39 type. The control will return error message *3036 G39 CODE IN G40* if command G39 is issued in state G40 or 3D compensation mode.

### 14.4.7 General Information on Tool Nose Radius Compensation

In offset mode (G41, G42), the control will always have to compute the compensation vectors between two interpolation blocks in the selected plane. In practice it may be necessary to program between two interpolation blocks in the selected plane a non-interpolation block or an interpolation outside of the selected plane. They may be
– functions (M, S, T)
– dwell (G4 P)
– interpolation outside of the selected plane (G18) G1 Y
– call of a subprogram (M98 P)
– setting or canceling special transformations (G50, G51, G50.1, G51.1, G68, G69).

☞ *Note*: Calling a subprogram some carefulness is needed. Unless the subprogram is beginning with a motion command in the assigned plane, the interpolation will be distorted.

The control will accept the programming of a **single** block of the above type between two interpolation blocks in the program, leaving the path of the tool unaffected:

```
...G18 G42 G91...
N110 G1 X140 Z50
N120 G4 P2
N130 Z60
...
```

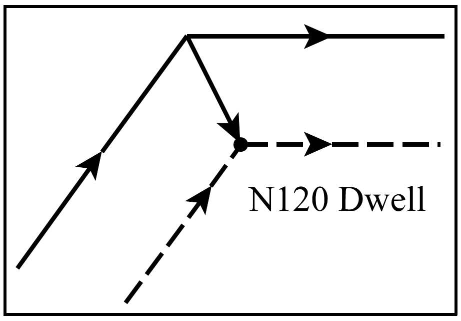


Fig. 14.4.7-1

*When the control inserts one or more straight lines between two interpolations when going around a corner, any other block without movement or with movement outside of the selected plane programmed between the interpolations will be executed at the single block stop point (indicated by "S" in the figures).*

When two interpolations outside of the selected plane or two blocks containing no interpolations are written in the program, the control will set an offset vector perpendicular to the end point of the last interpolation in the selected plane and the path will be distorted:

```
...G18 G42 G91...
N110 G1 X140 Z50
N120 G4 P2
N130 S400
N140 Z60
...
```

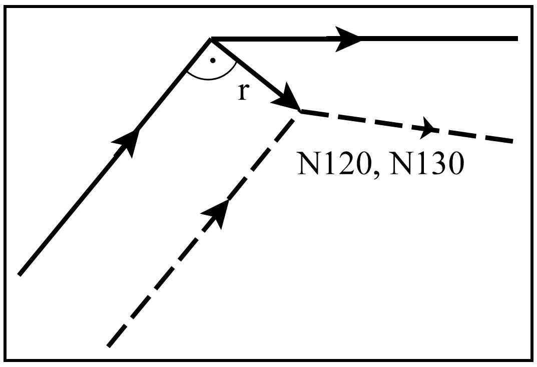


Fig. 14.4.7-2

If no cut is feasible in direction **Y** unless the radius compensation is set up, the following procedure may be adopted:

```
        ...G18 G91...
        N110 G41 G0 X140 Z50
        N120 G1 Y-40
        N130 X80
        ...
```

Now the tool will have a correct path as is shown in the Figure.

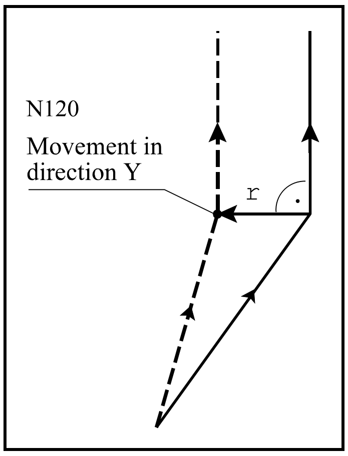


Fig. **14.4.7-3**

If, however, movement in direction **Y** is broken up into two sections (rapid traverse and feed), the path will be distorted because of the two consecutive interpolations outside of the selected plane:

```
        ...G18 G91...
        N110 G41 G0 X140 Z50
        N120 Y-35
        N130 G1 Y-5
        N140 X80
        ...
```

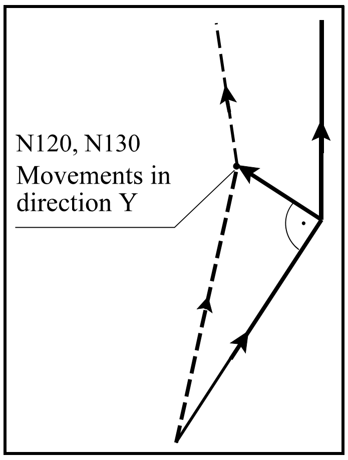


Fig. **14.4.7-4**

As a trade-off, insert a small movement in direction **X** between two ones in direction **Y**:

```
        ...G18 G91...
        N110 G41 G0 X1380 Z50
        N120 Y-35
        N130 X2
        N140 G1 Y-5
        N150 X80
        ...
```

With the above "trick" a correct compensation vector can be got.

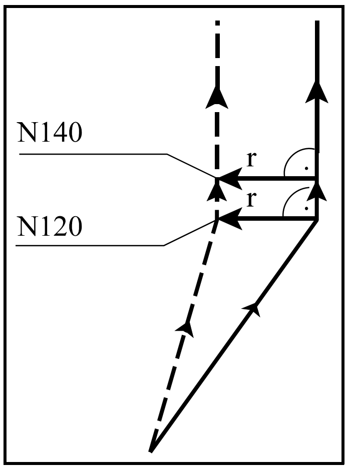


Fig. **14.4.7-5**

The path of tool will be as follows when instructions

G22, G23, G52, G54-G59, G92
G53
G28, G29, G30

are inserted between two interpolations.

When command G22, G23, G52, G54-G59 or G92 is programmed in offset mode between two interpolation blocks, the compensation vector will be deleted at the end point of the previous interpolation, the command will be executed and the vector will be restored at the end point of the next interpolation. If the previous or next interpolation is a circular one, the control will return error message *3041 AFTER G2, G3 ILLEG. BLOCK*.

For example:

```
...G91 G18 G41...
N110 G1 X–100 Z80
N120 G92 X0 Z0
N130 X100 Z80
...
```

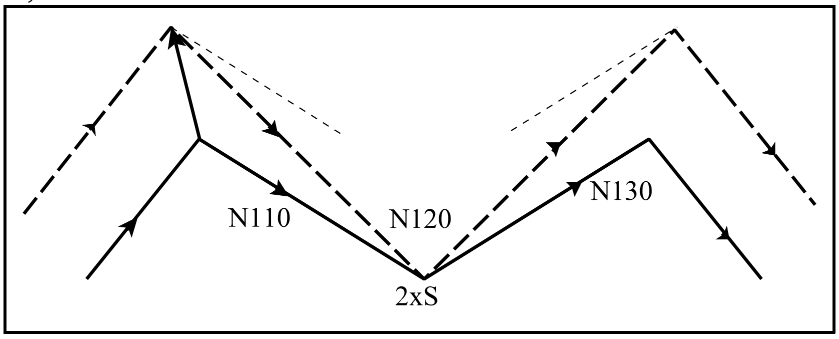


Fig. **14.4.7-6**

If command G53 is programmed in offset mode between two interpolations, the compensation vector will be deleted at the end point of the previous block, the positioning will be executed in G53, and the vector will be restored at the end point of the next interpolation (other than G53). If the previous or next interpolation is a circular one, the control will return error message *3041 AFTER G2, G3 ILLEG. BLOCK*.

For example:

```
...G91 G18 G41...
N110 G1 X–100 Z80
N120 G53 X1600
N130 G53 X0
N140 X100 Z80
...
```

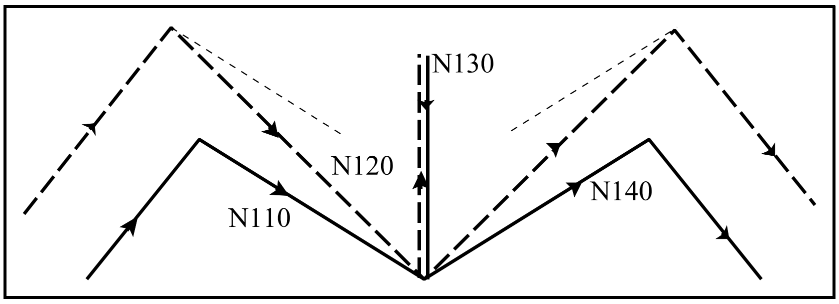


Fig. **14.4.7-7**

If G28 or G30 is programmed (followed by G29) between two blocks in offset mode, the compensation vector will be deleted at the end point of the block it positions the tool to the intermediate point, the tool will move to the reference point, and the vector will be restored at the end point of the returning block G29.

For example:

```
...G91 G18 G41...
N110 G1 X–100 Z80
N120 G28 X160
N130 G29 X0
N140 X100 Z80
...
```

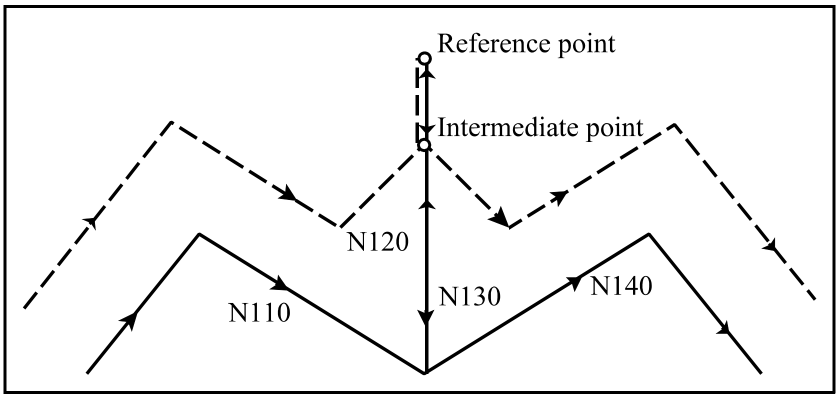


Fig. **14.4.7-8**

A new compensation value can also be called at address T in offset mode. In the event of a reversal in the sign of the radius, the direction of motion along the contours will be reversed (see earlier). Otherwise, the following procedure will be applicable. The compensation vector will be calculated with the new radius value at the end point of the interpolation, in which the new address T has been programmed. Since the compensation vector has been computed with the previous radius value at the start point of that block, the path of the tool center will not be parallel to the programmed path. A new radius compensation value can be called at address T in a circular interpolation, too, this time, however, the tool center will be moving along an arc with a variable radius.

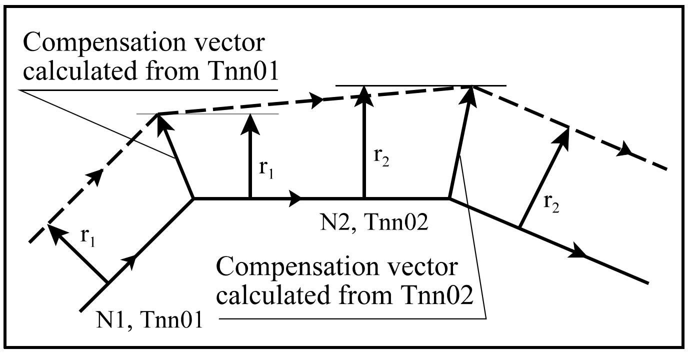

Compensation vector calculated from Tnn01
r1
r1
r2
r2
N2, Tnn02
Compensation vector calculated from Tnn02
N1, Tnn01

Fig. **14.4.7-9**

A special instance of the foregoing is canceling or setting up the compensation with T00 or Tnn, respectively, while in offset mode. Notice the difference in tool paths with reference to the following example, when the compensation is set up with G41 or G42 and canceled with G40, or when the compensation is set up and canceled by programming T.

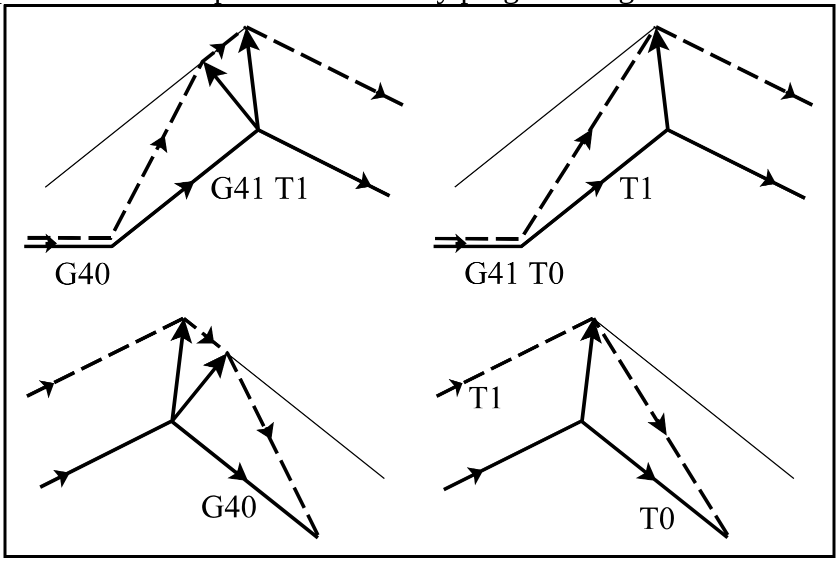


Fig. **14.4.7-10**

A particular program detail or subprogram may be used also for machining a male or female work-piece with positive or negative radius compensation, respectively, or vice-versa.

Let us review the following small program detail:

```
...
N020 G42 G1 X160 T1
N030 G1 Z-5
N040 G3 I-80
N050 G1 Z2
N060 G40 G0 X0
...
```

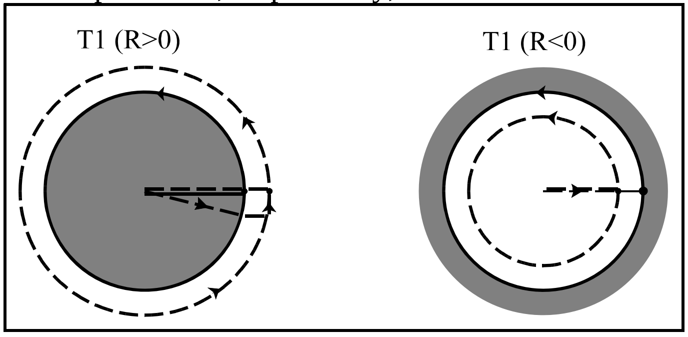


Fig. **14.4.7-11**

When the radius compensation is applied to a circle of a variable radius, the control will calculate the compensation vector(s) to an imaginary circle at the start point thereof, the radius of which is equal to the start-point radius of the programmed circle, the center point coinciding with the programmed one. The compensation vector(s) will be computed to an imaginary circle at the end point of it, the radius of which is equal to the end-point radius of the programmed circle, the center point coinciding with that of the programmed circle.

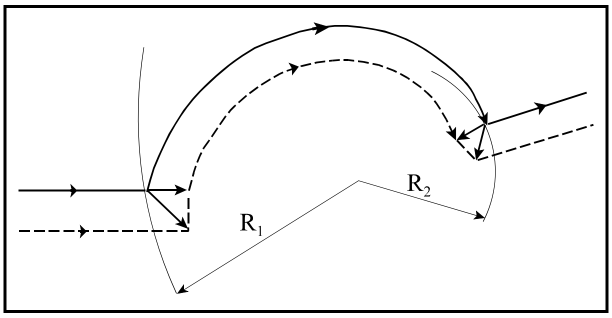


Fig. **14.4.7-12**

When a full circle is being programmed, it may often occur that the path of tool covers more than a complete revolution round the circle in offset mode.

For example, this may occur in programming a direction reversal along the contours:

```
...G17 G42 G91...
N110 G1 X30 Y-40
N120 G41 G2 J-40
N130 G42 G1 X30 Y40
...
```

The tool center covers a full arc of a circle from point $P_1$ to point $P_1$ and another one from point $P_1$ to point $P_2$.

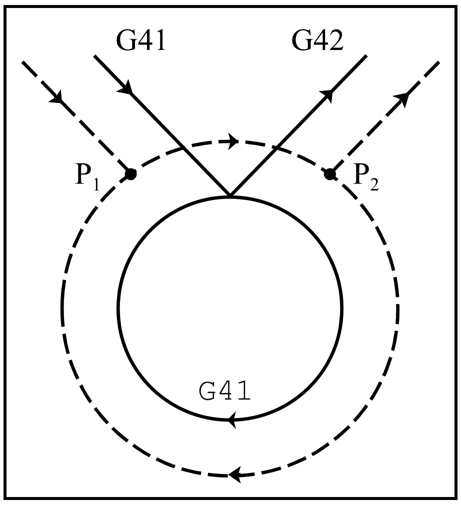


Fig. **14.4.7-13**

When offset mode is canceled by programming I, J, K, a similar condition will emerge:

```
...G18 G90 G41...
N090 G1 Z60
N100 G2 I-60
N110 G40 G1 X360 Z120 I-60 K-60
...
```

The tool center covers a full arc of a circle from point $P_1$ to point $P_1$ and another one from point $P_1$ to point $P_2$.

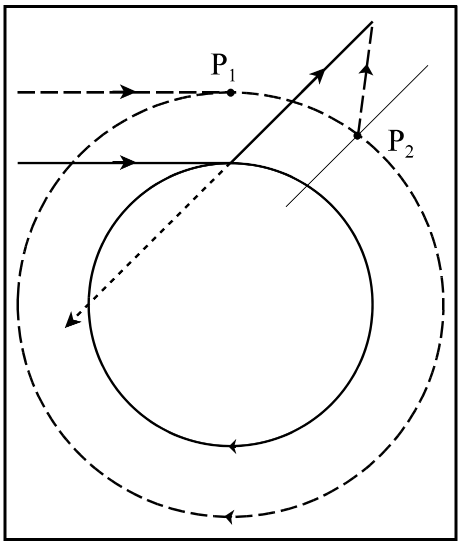


Fig. **14.4.7-14**

Two or more compensation vectors may be produced when going around sharp corners. When their end points lie close to each other, there will be hardly any motion between the two points.
When the distance between the two vectors is smaller than the value of parameter *DELTV* in each axis, the vector shown in the Figure will be omitted, and the path of the tool will be modified accordingly.

☞ *Note*: When parameter *DELTV* is too high (in causeless way) the sharp corners with acute angles may be overcut.

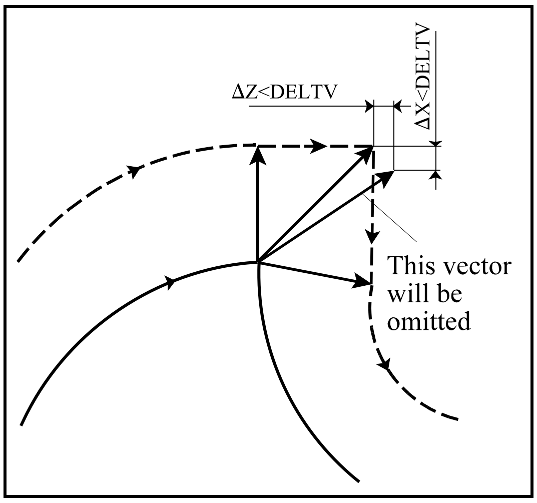


Fig. **14.4.7-15**

### 14.4.8 Interferences in Tool Nose Radius Compensation

It may frequently occur in offset mode that the path of the tool is the opposite of the programmed one. Under such conditions, the tool may cut into the workpiece contrary to the programmer's intentions. This phenomenon is referred to as the interference in tool nose radius compensation.

In the case shown in the Figure, after the intersection points have been computed, a tool path opposite to the programmed one will be obtained in the execution of interpolation N2. The hachure area indicates that the tool cuts in the workpiece.

To avoid this, the control performs an interference check when parameter *INTERFER* is set to 1. Now the control will check that the condition $-90° \le \varphi \le +90°$ is fulfilled for angle $\varphi$ between the programmed displacement and the one compensated with the radius.

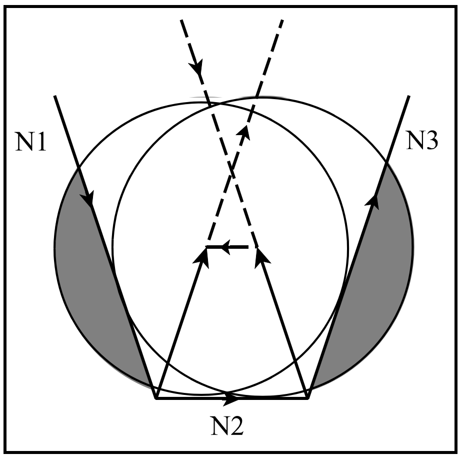


Fig. **14.4.8-1**

In the other words the control will check wether the compensated displacement vector has a component opposite to the programmed displacement vector or not.

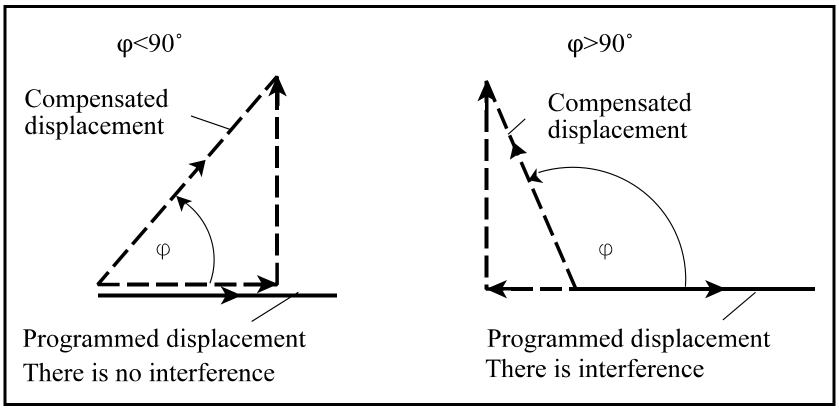


Fig. **14.4.8-2**

If parameter *ANGLAL* is set to 1, the control will, after an angle check, return an interference error message *3048 INTERFERENCE ALARM* one block earlier than the occurrence of the trouble.

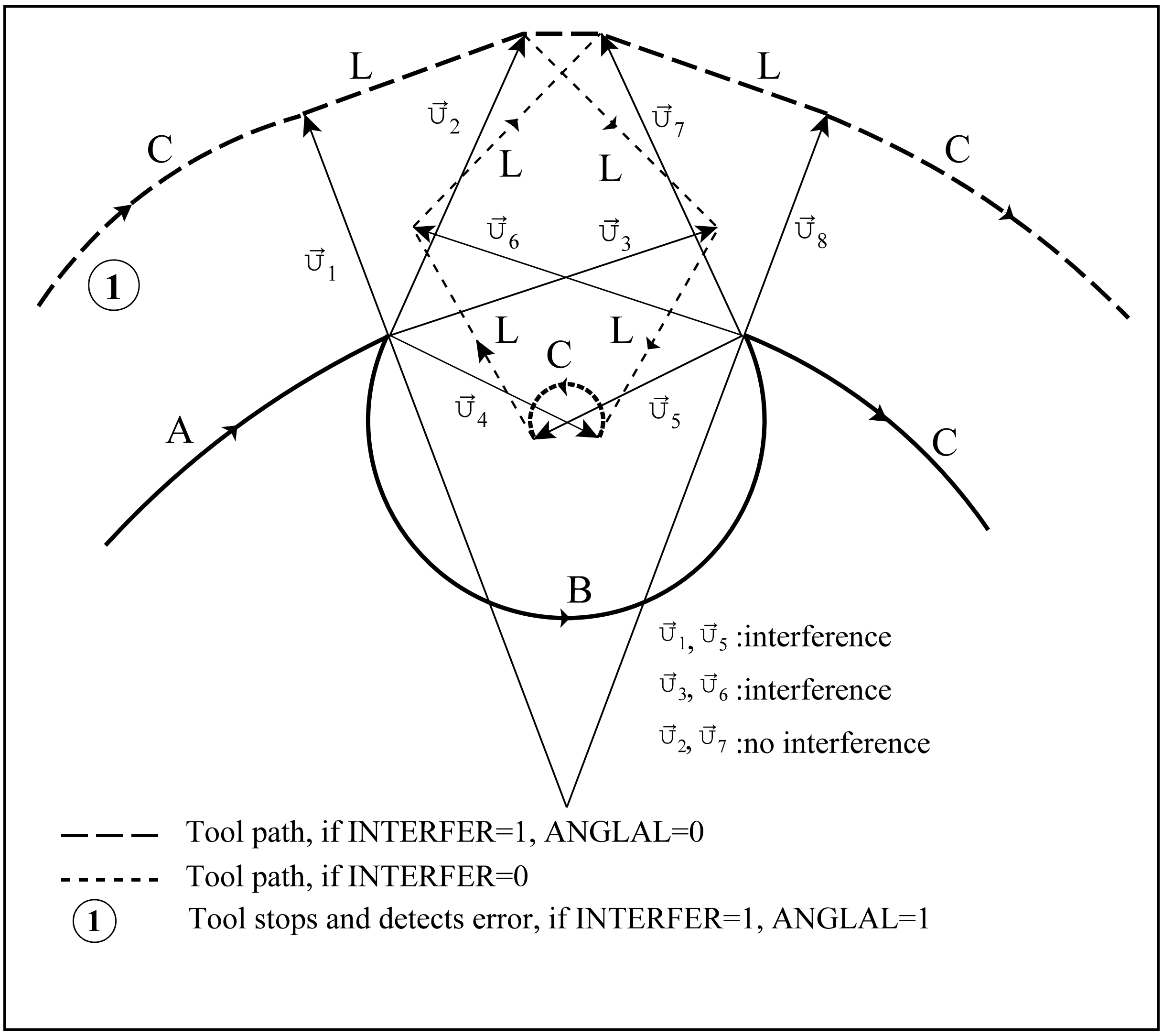

L
L
C
C
$\vec{u}_2$
$\vec{u}_7$
L
L
$\vec{u}_6$
$\vec{u}_3$
$\vec{u}_8$
$\vec{u}_1$
1
L
L
C
$\vec{u}_4$
$\vec{u}_5$
A
C
B
$\vec{u}_1$, $\vec{u}_5$ :interference
$\vec{u}_3$, $\vec{u}_6$ :interference
$\vec{u}_2$, $\vec{u}_7$ :no interference
Tool path, if INTERFER=1, ANGLAL=0
Tool path, if INTERFER=0
1 Tool stops and detects error, if INTERFER=1, ANGLAL=1

Fig. **14.4.8-3**

If parameter *ANGLAL* is set to 0, the control will not return an error message, but will automatically attempt to correct the contour in order to avoid overcutting. The procedure of compensation

is as follows.
Each of blocks A, B and C are in offset mode. The computed vectors between blocks A and B are $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$, $\vec{v}_4$; the compensation vectors between blocks B and C are $\vec{v}_5$, $\vec{v}_6$, $\vec{v}_7$, $\vec{v}_8$.

- $\vec{v}_4$ and $\vec{v}_5$ will be ignored if there is an interference between them.
- $\vec{v}_3$ and $\vec{v}_6$ will be ignored if there is an interference between them.
- $\vec{v}_2$ and $\vec{v}_7$ will be ignored if there is an interference between them.
- $\vec{v}_1$ and $\vec{v}_8$ cannot be omitted in the case of an interference, so an error message is returned.

It is evident from the foregoing that the compensation vectors are paired at the start and end points of interpolation B, and will be ignored in pairs. If the number of compensation vectors on one side is 1 (or is reduced to 1), only the vectors on the other side will be omitted. The procedure of omitting will be carried on as long as the interference persists. The first compensation vector at the start point of interpolation B and the last one at the respective end point cannot beignored. If, as a result of omissions, the interference is eliminated, no error message will be returned, but error message *3048 INTERFERENCE ALARM* will be returned otherwise. The remaining compensation vectors after each omission will always be interconnected by straight lines - even if interpolation B has been a circular one.
It is evident from the above example that the execution of interpolation A will not be commenced unless interpolation B has been checked for an interference. To do so, however, block C also had to be entered in the buffer, and the compensation vectors had to be calculated for transition B - C.

A few typical instances of interference will be described below.

Cutting a step smaller than the tool radius. The control returns error message *3048 INTERFERENCE ALARM* or else it would cut in the workpiece.

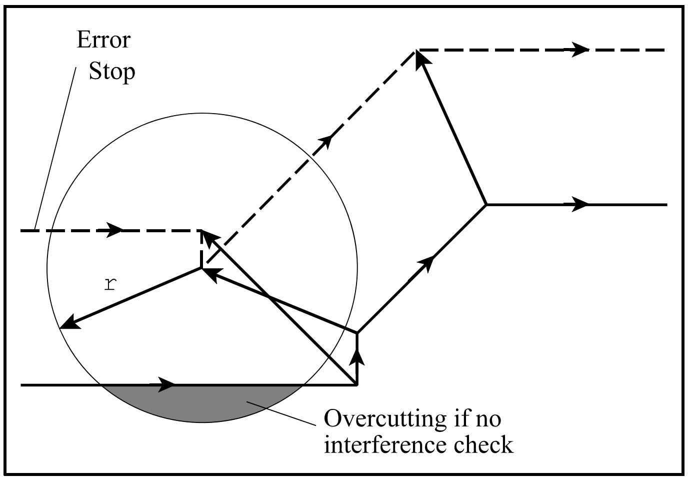


Fig. **14.4.8-4**

Machining an inside corner with a radius smaller than the tool radius. The control returns error message *3048 INTERFERENCE ALARM* or else overcutting would occur.

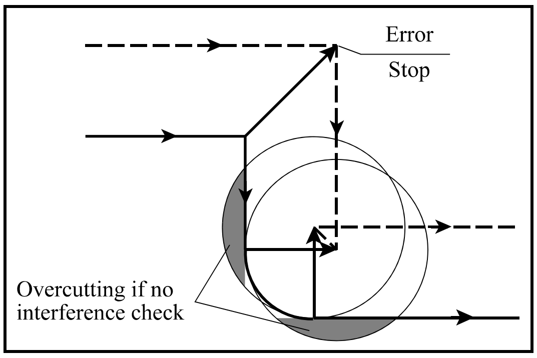


Fig. **14.4.8-5**

Cutting a step smaller than the tool radius along an arc. If parameter *ANGLAL* is 0, the control will delete vector $\vec{\upsilon}_2$ and will interconnect vectors $\vec{\upsilon}_1$ and $\vec{\upsilon}_3$ by a straight line to avoid a cut-in. If parameter ANGLAL is 1 it returns error message *3048 INTERFERENCE ALARM* and stops at the end of previous block.

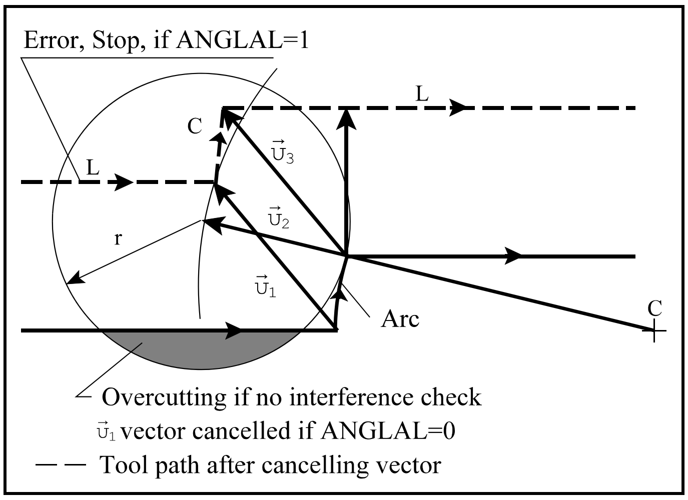

Error, Stop, if ANGLAL=1
L
C
L
$\vec{\upsilon}_3$
$\vec{\upsilon}_2$
$\vec{\upsilon}_1$
r
Arc
C
Overcutting if no interference check
$\vec{\upsilon}_1$ vector cancelled if ANGLAL=0
— — Tool path after cancelling vector

Fig. **14.4.8-6**

Sometimes the tool would not actually overcut the workpiece, but the interference check indicates an error.

If a recess smaller than the radius compensation is being machined, actually no overcut would occur (see the Figure), yet the control returns error message *3048 INTERFERENCE ALARM* because the direction of displacement along the compensated path in interpolation B is opposite to the programmed one.

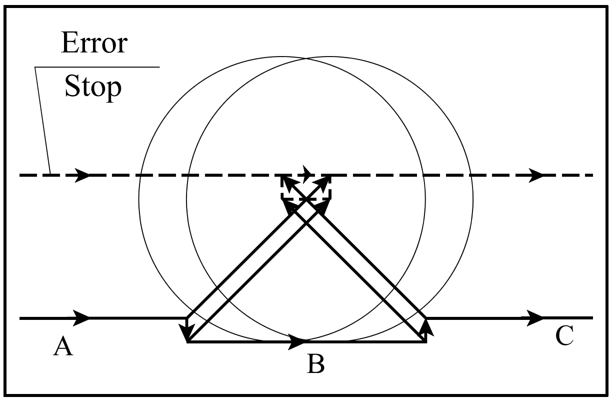


Fig. **14.4.8-7**

In the above example an interference error is returned again because the displacement of the compensated path in interpolation B is opposite to the programmed one.

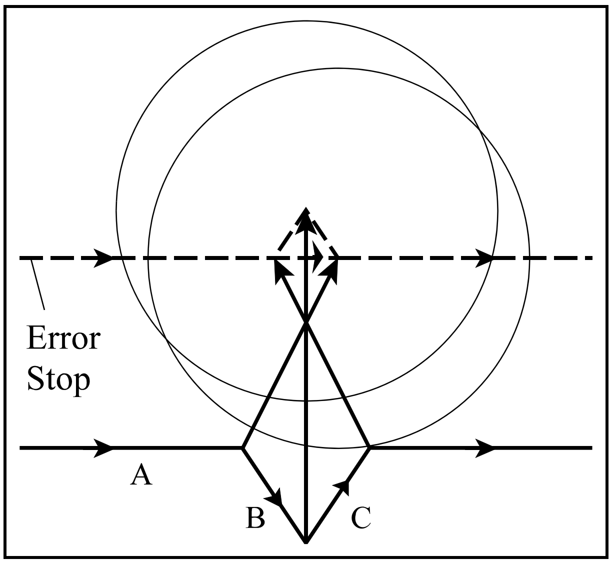


Fig. **14.4.8-8**

# 15 Special Transformations

## 15.1 Mirror Image for Double Turret (G68)

Command

**G68** switches double turret mirror image on, while command

**G69** cancels it.

This function can be used for the programming of two facing turrets or tool posts. The first tool post, tool post “A” machines in the positive quadrant, while the second one, tool post “B” machines in the negative quadrant. Mirror image is done always to X axis as the effect of command G68. In this case coordinates programmed to X axis change sign, at the same time coordinate transformation occurs along X axis. The coordinate offset is determined by the distance between the two turrets, which can be specified at parameter 1001 DTPX. Afterwards the program must be written as if the original turret, turret “A” is applied, i.e., it must be written for the positive quadrant. For example:

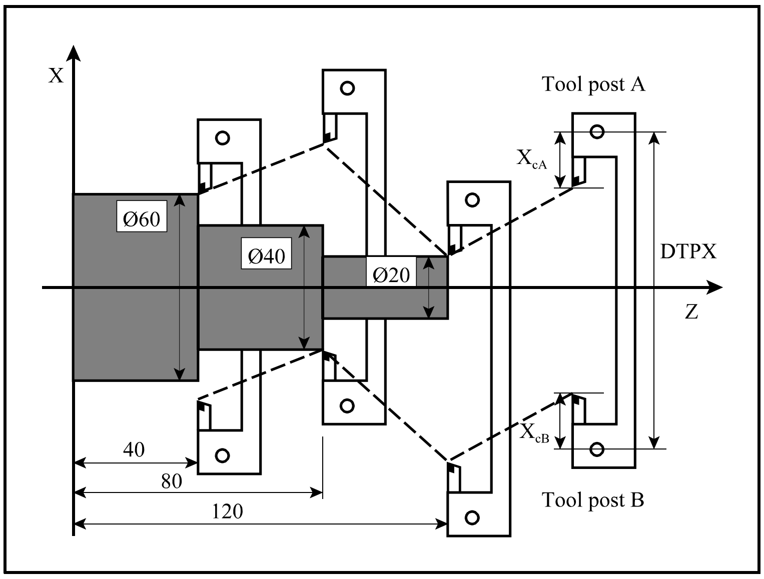


Fig. **15.1**-1

```
T101        (tool No. 1 in turret “A” is activated)
G0 X20 Z120 (tool No. 1 is positioned)
G68         (mirror image is canceled)
T202        (tool No. 2 in turret “B” is activated)
G0 X40 Z80  (tool No. 2 is positioned)
G69         (mirror image is canceled)
T101        (tool No. 1 in turret “A” is activated)
G0 X60 Z40  (tool No. 1 is positioned)
```

*Note:*

–*Commands G68 and G69 must always be in single block, other commands cannot be programmed in that block.*

–*In order to apply the function the distance between the two turrets must be measured and written at parameter 1001 DTPX.*

–*If G68 is on the following elements are inversed considering the programmed ones:*

*Sign of X coordinate,*

*circle direction G2 changes to G3 and reversely,*

*direction G41 changes to G42 and reversely.*

## 15.2 Scaling (G50, G51)

Command

**G51** v P

can be used for scaling a programmed shape.

| | |
|---|---|
| P1...P4: | Points specified in the part program |
| P1'...P4': | Points after scaling |
| P0: | Center of scaling |

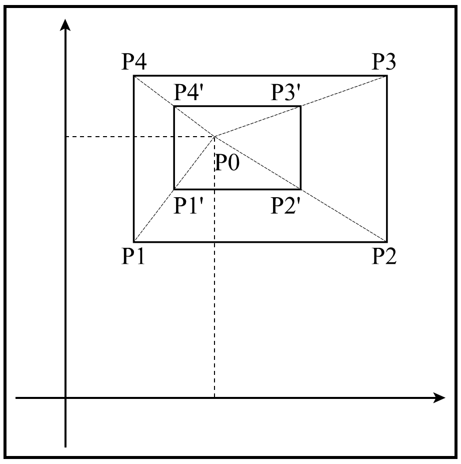


Fig. **15.2-1**

The coordinates of the scaling center can be entered at coordinates of v. The applicable addresses are X, Y, Z, U, V, W. The coordinate data of v entered here will also be interpreted as rectangular (Cartesian) data, even when the polar coordinate data specification is set up.

Using G90, G91 or operator I, the v coordinates of the center of scaling can be specified as absolute or incremental data.

Unless one or both axes' addresses are assigned values, the instantaneous axis position will be taken for the center of scaling.

The scale factor can be specified at address P. Its value can be represented by 8 decimal digits; the position of the decimal point is irrelevant.

Scaling can be canceled with command

**G50**.

For example:

```
N1 G90 G0 X100 Z120
N2 G51 X0 Z0 P0.5
N3 G1 X0 Z100 F150
N4 X80
N5 Z0
N6 G50
N7 G0 X100 Z120
```

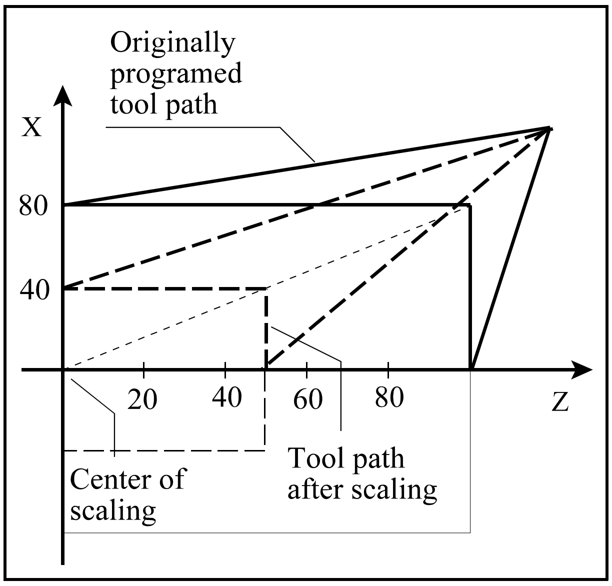


Fig. **15.2-2**

## 15.3 Programmable Mirror Image (G50.1, G51.1)

A programmed shape can be projected as a mirror image along the coordinates selected in v by command

**G51.1** v

in such a way that the coordinates of the axis (or axes) of mirror image can be specified in v. The v coordinate may be X, Y, Z, U, V, W, A, B, C.

The v coordinate data entered here are interpreted as rectangular coordinate data even when polar coordinate data specifications are set up.

Using G90, G91 or operator I, the v coordinates of the axes of the mirror image can be specified as absolute or incremental data.
No mirror image will be on the axis, for the address of which no value has been assigned.

Command

**G50.1** v

will cancel the mirror image on axis (axes) specified at v. Any arbitrary data can be written for the v coordinates, its effect will only record the fact of canceling.
When this command is issued, no rotation or scaling command may be in effect. Otherwise an error message *3000 MIRROR IMAGE IN G51, G68* is returned.
When a mirror image is applied on an axis of composing the selected plane:
– The circle direction is reversed automatically (interchange of G02, G03)
– The angle of rotation is assigned an opposite meaning (G68).

Example:

Subprogram

```
O0101
N1 G0 X40 F120
N2 G1 Z80
N3 G3 X80 Z100 R20
N4 G1 X100 Z110
N5 M99
```

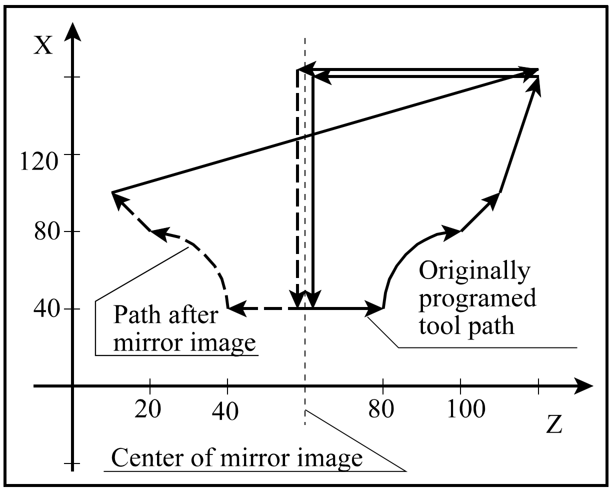


Fig. **15.3**-1

main program

```
O0100
N10 T101
N20 G0 X160 Z60 M3 S1000
N30 M98 P101       (subprogram call)
N40 G0 X160 Z120
N50 T202
N60 G0 X160 Z60 M3 S1000
N70 G51.1 Z60      (mirror image parallel to X, coordinate Z=60)
N80 M98 P101       (subprogram call)
N90 G50.1 Z0       (canceling mirror image parallel to X)
N100 G0 X160 Z120
N110 M30
```

The mirror image can only be switched on in state G50, i.e., if the scaling command is not active. However, in the switched-on state of the mirror image the scaling can also be switched on. It is also true for the mirror image that it cannot be overlapped with scaling command, i.e., first the scaling has to be canceled and only then the cancellation of the mirror image may come:

```
G51.1 ...          (mirror image on)
G51 ...            (scaling on)
...
G50 ...            (scaling off)
G50.1 ...          (mirror image off)
```

# 16 Automatic Geometric Calculations

## 16.1 Programming Chamfer and Corner Round

The control is able to insert chamfer or rounding between two blocks containing linear (G01) or circle interpolation (G02, G03) automatically.

A chamfer, the length of which equals to the value specified at address

,C

(comma and C) is inserted between the end point of the block containing address ,C and the start point of the forthcoming block. E.g.:

```
N1 G1 G91 Z30 ,C10
N2 X80 Z10
```

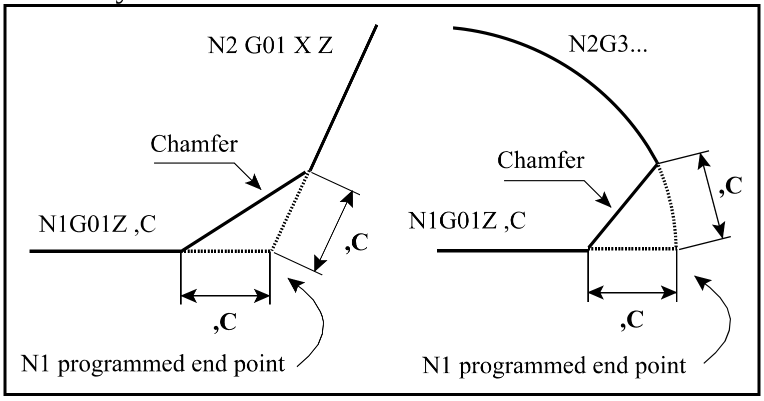


Fig. **16.1**-1

The value specified at the address ,C shows the distance between the start/end point of chamfer and the supposed intersection of the two successive blocks. A chamfer may also be inserted between circles or between a circle and a straight line. In this case value ,C is the length of the chord drawn from intersection.

A rounding, the radius of which corresponds to the value given at address

,R

(comma and R) is inserted between the end point of the block containing address ,R and the start point of the forthcoming block. E.g.:

```
        N1 G91 G01 Z30 ,R8
        N2 G03 X60 Z–30 R30
```

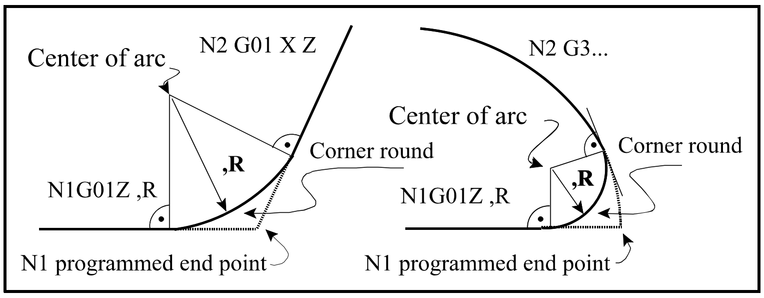


Fig. **16.1**-2

A ,R-radius arc is inserted between the two blocks so that the circle osculates to both path elements. Command containing a chamfer or a corner round ing may also be written at the end of more successive blocks as shown in the below example:

```
        ...
        G1 X80 ,C10
        Z60 ,R22
        G3 X160 Z20 R40 ,C10
        G1 X220
        ...
```

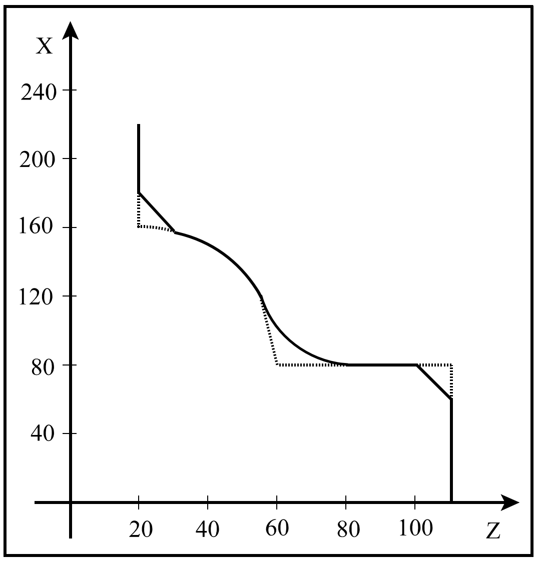


Fig. **16.1**-3

☞ *Note:*
– Chamfer or rounding can only be programmed between the coordinates of the selected plane (G17, G18, G19), otherwise error message *3081 DEFINITION ERROR ,C ,R* is sent by the control.
– Chamfer or corner rounding can only be applied between blocks G1, G2 or G3, otherwise error message *3081 ,C ,R DEFINITION ERROR* is sent by the control
– Provided the length of chamfer or the rounding radius is so great, that it cannot be inserted to the programmed blocks, error message *3084 ,C ,R TOO HIGH* is sent by the control.
– If both ,C and ,R are programmed within one block error message *3017,C AND ,R IN ONE BLOCK* is sent by the control.
– In single block mode the control stops and registers STOP state after the execution of chamfer or corner rounding.

## 16.2 Specifying Straight Line with Angle

Straight line can be specified in the plane determined by commands G17, G18, G19 by means of a coordinate of the selected plane and the angle given at address ,A.

$$G17 \left\{\begin{matrix} G00 \\ \hline G01 \end{matrix}\right\} \left\{\begin{matrix} X_p\ ,A \\ \hline Y_p\ ,A \end{matrix}\right\} q\ F$$

$$G18 \left\{\begin{matrix} G00 \\ \hline G01 \end{matrix}\right\} \left\{\begin{matrix} Z_p\ ,A \\ \hline X_p\ ,A \end{matrix}\right\} q\ F$$

$$G19 \left\{\begin{matrix} G00 \\ \hline G01 \end{matrix}\right\} \left\{\begin{matrix} Y_p\ ,A \\ \hline Z_p\ ,A \end{matrix}\right\} q\ F$$

In the above formulas Xp, Yp, Zp indicate X, Y, Z axes or those parallel to them, while q represents an optional axis out of the selected plane. Specification at address ,A can also be used b-eside codes G0 and G1. The angle is measured from the first axis of the selected plane and the positive direction is counter-clockwise. Value ,A may be both positive or negative as well as greater than 360° or less than −360°.

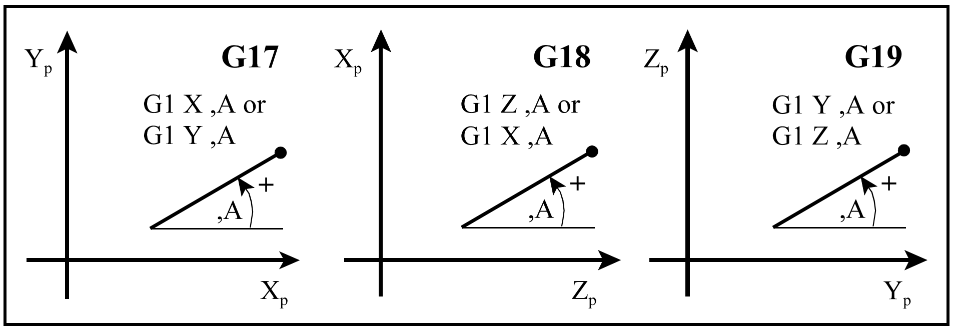


Fig. **16.2-1**

☞*Note:*

– *In case the coordinate system is situated like on the figure beside the interpretation of angle is modified as seen in the enclosed figure (positive direction is clockwise).*

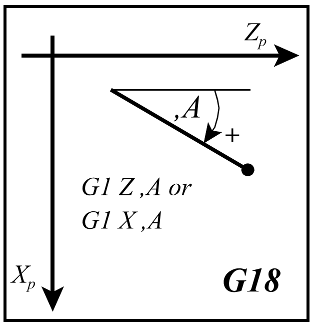


Fig. **16.2**-2

For example:

```
(G18 G90) G1 X60 Z120 ...
Z70 ,A150   (this specification is equi-
            valent to X117.735 Z70)
X180 ,A135  (this specification is equi-
            valent to X180 Z38.868)
```

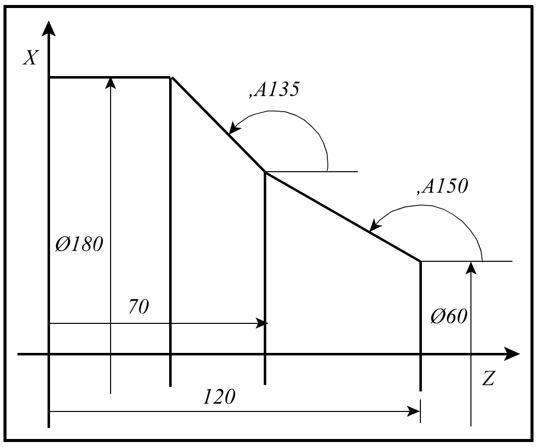


Fig. **16.2**-3

☞*Note:*

– Straight line with angle together with chamfer or corner rounding can be defined in one block. For example:

```
Z100 ,A30 ,C5
X100 ,A120 ,R10
Z-100 ,A210
```

– Angle specification at address ,A can also be applied in drilling cycles. In this case it is acknowledged in the course of positioning execution in the selected plane.

For example block

```
G81 G91 X100 ,A30 R-2 Z-25
```

is equivalent to the block below:

```
G81 G91 X100 Y57.735 R-2 Z-25
```

## 16.3 Intersection Calculations in the Selected Plane

Intersection calculations discussed here are only executed by the control when ***tool radius compensation (G41 or G42 offset mode) is on***. If eventually no tool radius compensation is needed in the part program turn the compensation on and use 0 offset for the tool radius.

### 16.3.1 Linear-linear Intersection

If the second one of two successive linear interpolation blocks is specified the way that its both end point coordinates in the selected plane and also its angle is specified, the control calculates the intersection of the straight lines referred to in the first block and the straight line specified in the second one. The straight line specified in the second block is determined over. The calculated intersection is the end point of the first block, as well as the start point of the second one.

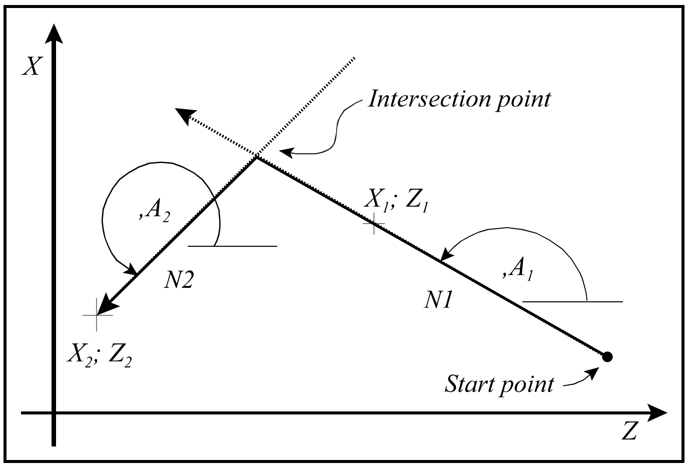


Fig. **16.3.1-1**

| | | |
|---|---|---|
| G17 G41 (G42) | G18 | |
| N1 G1 ,A1 or | G41 (G42) | G19 G41 (G42) |
| X1 Y1 | N1 G1 ,A1 or | N1 G1 ,A1 or |
| N2 G1G90 **X2 Y2 ,A2** | X1 Z1 | Y1 Z1 |
| | N2 G1G90 **X2 Z2 ,A2** | N2 G1G90 **Y2 Z2 ,A2** |

The intersection is always calculated in the plane selected by G17, G18, G19. The first block (N1) is specified either by means of its angle (,A1), in this case a straight line is drawn from the start point to the intersection point in the appropriate angle, or with an optional position other than the start point of the straight line (X1, Y1; X1, Z1; or Y1, Z1). In this case the intersection is calculated with the straight line lying on both points. Coordinates given 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
...
```

Block N10 can also be given with 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
...
```

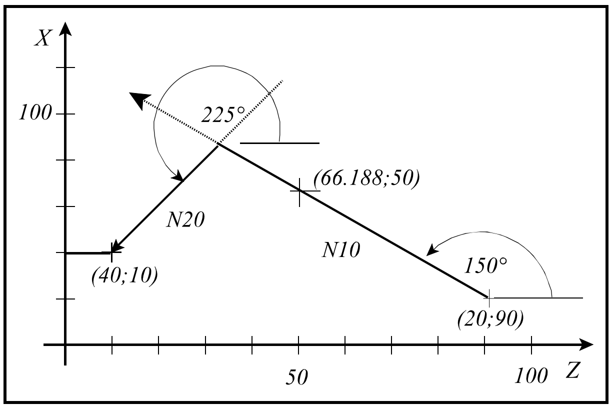


Fig. 16.3.1-2

Note, that in this case coordinate X, Z (X66.18 Z50) given in block N10 is not acknowledged by the control as end point, but as a transit position binding the straight line with the start point.
Intersection calculation can also be combined with a chamfer or corner rounding specification.
E.g.:

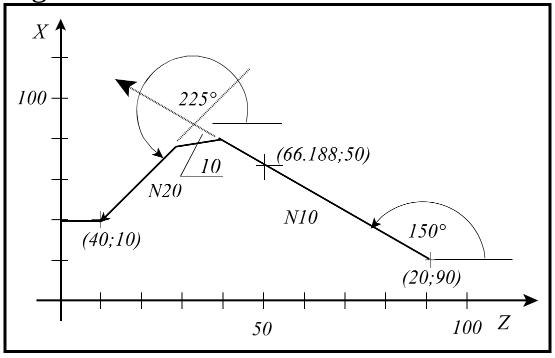


Fig. 16.3.1-3

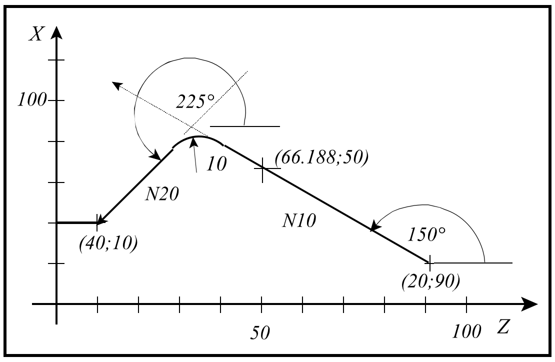


Fig. 16.3.1-4

```
(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 above examples chamfering amount is measured from the calculated intersection, as well as rounding is inserted to the calculated intersection.

## 16.3.2 Linear-circular Intersection

If a circular block is given after a linear block in a way that the end and center position coordinates as well as the radius of the circle are specified, i.e., the circle is determined over, then the control calculates intersection between straight line and circle. The calculated intersection is the end point of the first block, as well as the start point of the second one.

```
G17 G41 (G42)
N1 G1 ,A or
        X1 Y1
N2 G2 (G3) G90 X2 Y2 I J
R Q
```

```
G18 G41 (G42)
N1 G1,A or
        X1 Z1
N2 G2 (G3) G90 X2 Z2 I
K R Q
```

```
G19 G41 (G42)
N1 G1 ,A or
        Y1 Z1
N2 G2 (G3) G90 Y2 Z2 J
K R Q
```

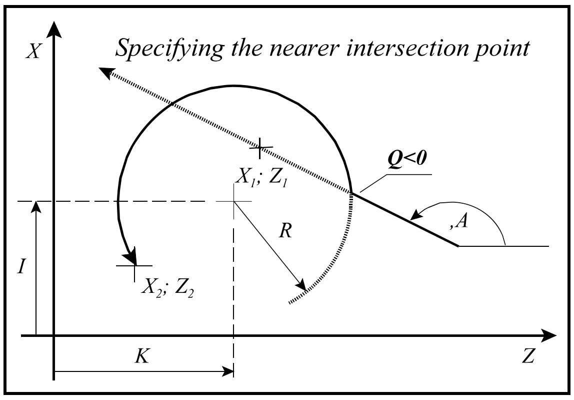


Fig. 16.3.2-1

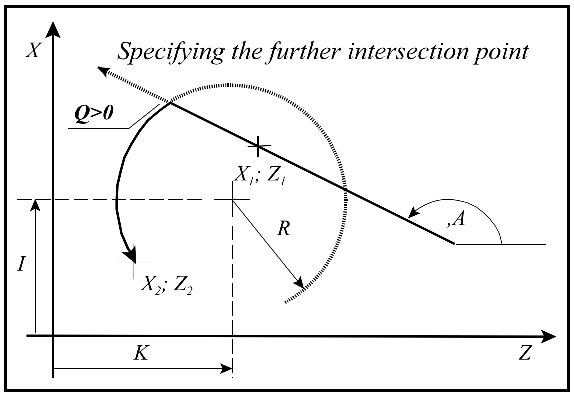


Fig. 16.3.2-2

The intersection is always calculated in the plane selected by G17, G18, G19. The first block (N1) is specified either by means of its angle (,A), in this case a straight line is drawn from the start point to the intersection point in the appropriate angle, or an optional point other than the start point of the straight line is given ($X_1$, $Y_1$; $X_1$, $Z_1$; or $Y_1$, $Z_1$). In this case the intersection is calculated with the straight line lying on both points. Coordinates given in the second block (N2) also **I, J, K coordinates** defining the **center of the arc,** are always interpreted by the control as **absolute** data (G90). Of the two resulting intersections the one to be calculated by the control can be specified at address Q.

**If the address value is less than zero (Q<0) the nearer intersection in direction of the straight line is calculated, while if the address value is greater than zero (Q>0) the farther one in direction of the straight line is calculated. The direction of movement along the straight line is determined by the angle.**

Let us see the following example:

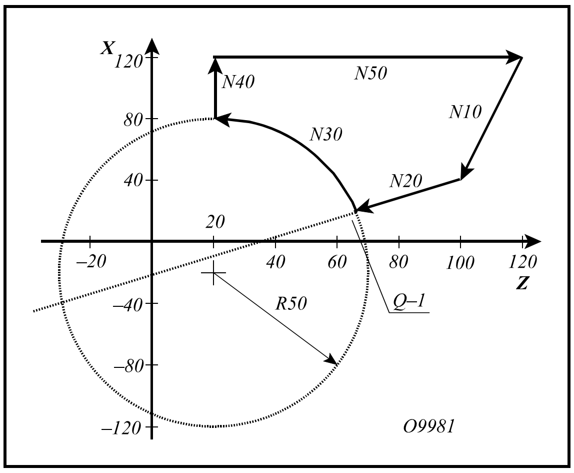


Fig. **16.3.2-3**

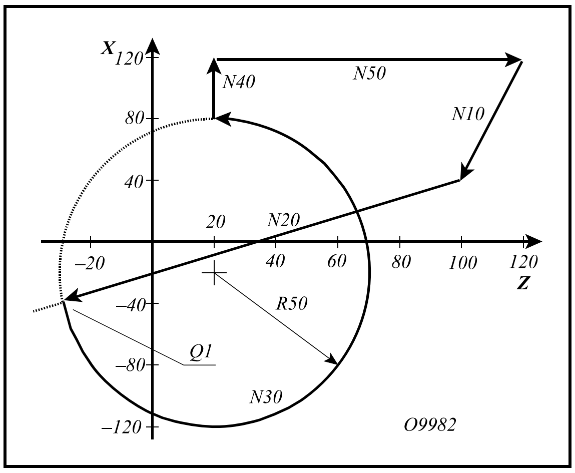


Fig. **16.3.2-4**

```
%O9981
N10 (G18) G42 G0 X40 Z100 S200 M3
N20 G1 X-40 Z-30
N30 G3 X80 Z20 I-10 K20 R50 Q-1
N40 G40 G0 X120
N50 Z120
N60 M30
%
```

```
%O9982
N10 (G18) G42 G0 X40 Z100 S200 M3
N20 G1 X-40 Z-30
N30 G3 X80 Z20 I-10 K20 R50 Q1
N40 G40 G0 X120
N50 Z120
N60 M30
%
```

Circular block N30 G3 is determined over, for both the center coordinates (I–10 K20 in absolute value) and the circle radius (R50) are specified, the control calculates the intersection of the straight line given in block N20 and the circle given in block N30. In program O9981 the nearer intersection in direction of the straight line is calculated because Q–1 is given in circular block N30.

Linear-circular intersection calculation can also be combined with chamfering or rounding specification. E.g.:

```
      %O9983
      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
      N60 M30
      %
```

The control calculates the intersection of blocks N20 and N30 and inserts a 15mm corner rounding as the effect of ,R15 given in block N20.

### 16.3.3 Circular-linear Intersection

If a linear block is given after a circular block in a way that the straight line is defined over, i.e., both its end point coordinate and the angle are specified, then the control calculates intersection between the circle and the straight line. The calculated intersection is the end point of the first block, as well as the start point of the second one.

```
G17 G41 (G42)                G18 G41 (G42)                G19 G41 (G42)
N1 G2 (G3) X1 Y1 I J         N1 G2 (G3) X1 Z1 I K         N1 G2 (G3) Y1 Z1 J K
      or R                             or R                          or R
N2 G1 G90 X2 Y2 ,A Q         N2 G1 G90 X2 Z2 ,A Q         N2 G1 G90 Y2 Z2 ,A Q
```

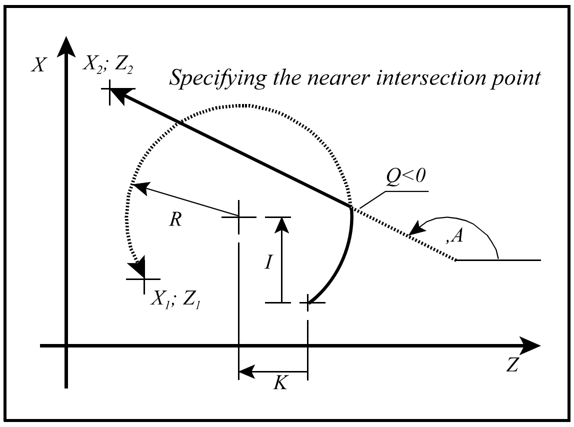


Fig. **16.3.3-1**

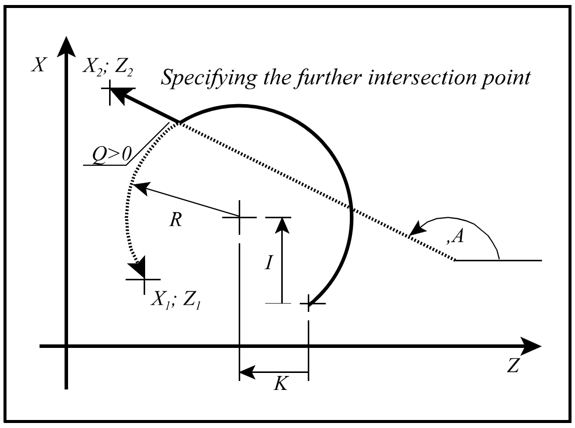


Fig. **16.3.3-2**

The intersection is always calculated in the plane selected by G17, G18, G19. The first block (N1), i.e., the circle is specified with an optional point ($X_1$, $Y_1$; $X_1$, $Z_1$; or $Y_1$, $Z_1$) and its center point coordinates (I J; I K; or J K) or instead of the center point coordinates with its radius (R). In the second block (N2) the straight line is determined over, i.e., both the end point coordinates ($X_2$ $Y_2$; $X_2$ $Z_2$; or $Y_2$ $Z_2$) and the angle (,A) of the straight line are given. The end point coordinates of the straight line are always interpreted by the control as **absolute** data (G90). It is always the **vector angle** of the straight line pointing from the resulting **intersection** at the given **end point** to be specified at address ,A, otherwise movements contrary to programmer's needs occur. Of the two resulting intersections the one to be calculated by the control can be specified at address Q.

**If the address value is less than zero (Q<0, e.g., Q–1) the nearer intersection in direction of the straight line is calculated, while if the address value is greater than zero (Q>0, e.g., Q1) the farther one in direction of the straight line is calculated. The direction of movement along the straight line is determined by the angle.**

Let us see an example:

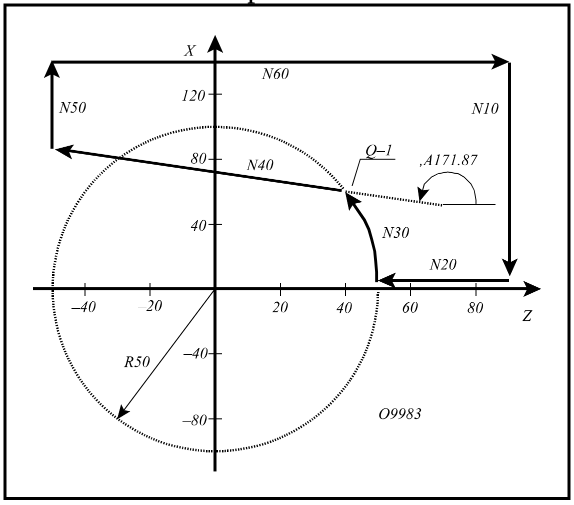


Fig. **16.3.3-3**

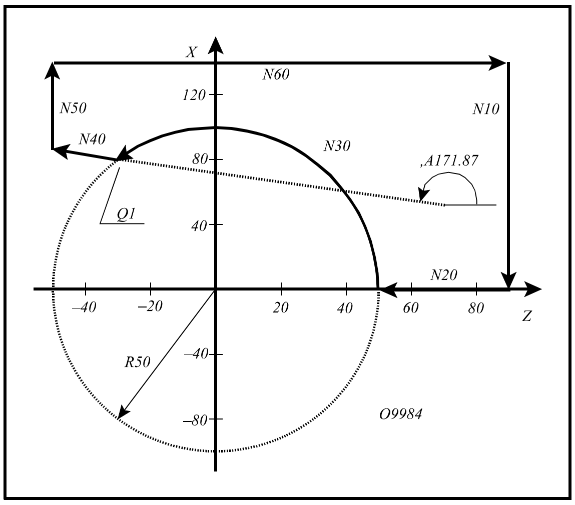


Fig. **16.3.3-4**

```
%O9983
N10 (G18) G0 X90 X0 M3 S200
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
N70 M30
%
```

```
%O9984
N10 (G18) G0 X90 X0 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
N70 M30
%
```

Linear block N40 is defined over because both the end point coordinates (X85.714 Z–50) and the angle (,A171.87) of the straight line are specified. Therefore X0 Z–50 coordinates of the circle programmed in the previous block N30 are not referred to as end point coordinates, but only as a point which is intersected by the circle and the end point is the calculated intersection. In program No. O9983 the nearer intersection in the direction of the straight line is given (Q–1), while in O9984 the farther one is specified (Q1).

Circular-linear intersection calculation can also be combined with a chamfer or rounding specification. E.g.:

```
%O9983
N10 (G18) G0 X90 X0 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
N70 M30
%
```

In the example a 15mm-rounding is programmed in block N30 (,R15). The control calculates the intersection of blocks N30 and N40 and inserts the programmed rounding to the resulting contour.

### 16.3.4 Circular-circular Intersection

If two successive circular blocks are specified so that the end point, the center coordinates as well as the radius of the second block are given, i.e., it is determined over the control calculates intersection between the two circles. The calculated intersection is the end point of the first block, as well as the start point of the second one.

```
G17 G41 (G42)
N1 G2 (G3) X1 Y1 I1 J1
        or X1 Y1 R1
N2 G2 (G3) G90 X2 Y2 I2
J2 R2 Q
```

```
G18 G41 (G42)
N1 G2 (G3) X1 Z1 I1 K1
        or X1 Z1 R1
N2 G2 (G3) G90 X2 Z2 I2
K2 R2 Q
```

```
G19 G41 (G42)
N1 G2 (G3) Y1 Z1 J1 K1
        or Y1 Z1 R1
N2 G2 (G3) G90 Y2 Z2 J2
K2 R2 Q
```

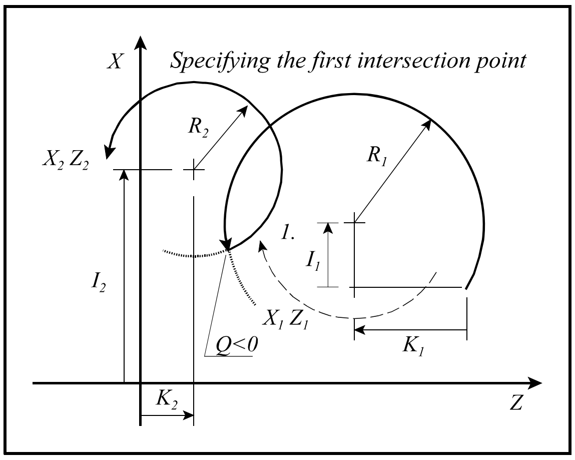


Fig. **16.3.4-1**

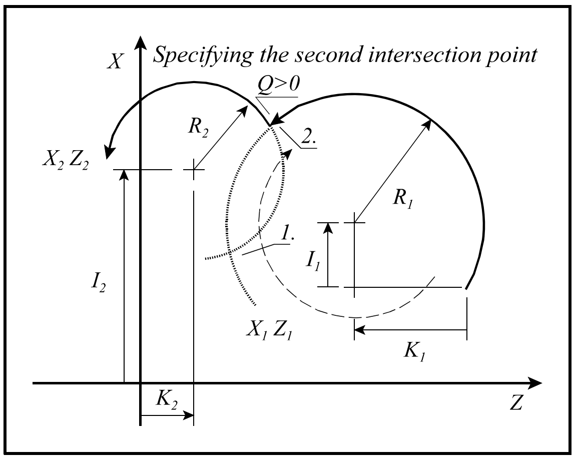


Fig. **16.3.4-2**

The intersection is always calculated in the plane selected by G17, G18, G19. The first block (N1) is specified either with the center coordinates ($I_1$ $J_1$; $I_1$ $K_1$; $J_1$ $K_1$) or with the radius ($R_1$) of the circle. In this block the interpretation of center coordinates corresponds to the default circle specification, i.e., it is the relative distance from the start point. The coordinates given in the second block (N2), as **I, J, K coordinates** defining the **circle center,** are always interpreted by the control as **absolute** data (G90). Of the two resulting intersections the one to be calculated by the control can be specified at address Q. If the address value is less than zero (Q<0, e.g., Q–1) the first intersection is calculated, while if the address value is greater than zero (Q>0, e.g., Q1) it is the second one.

**The first intersection is the one, first intersected going clockwise (independent of the programmed direction G2, G3).**

Let us see the following example:

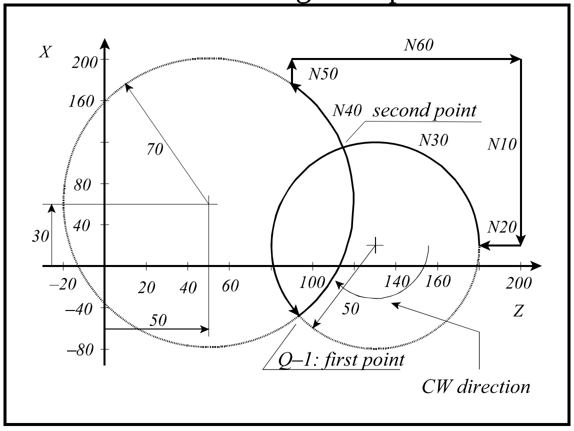


Fig. **16.3.4-3**

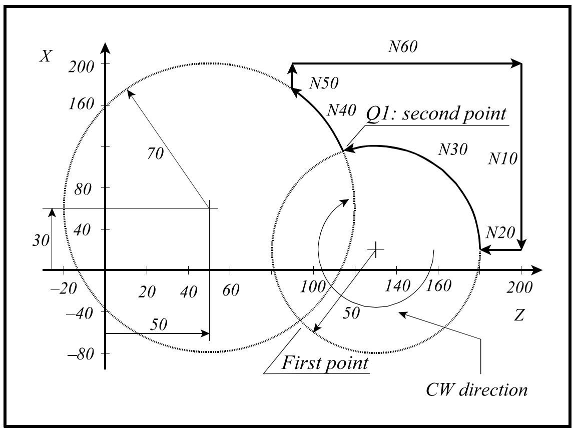


Fig. **16.3.4-4**

```
%O9985
N10 (G18) G0 X20 Z200 M3 S200
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
N70 M30
%
```

```
%O9986
N10 (G18) G0 X20 Z200 M3 S200
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
N70 M30
%
```

Circular block N40 is defined over because both, the center coordinates (I30 K50 in absolute value) and the radius (R70) of the circle, are specified. Therefore coordinates X–80 Z130 of the circle programmed in the previous block N30, are not referred to as end point coordinates, but only as a point which is lying on the circle and the end point is the calculated intersection. In program No. O9985 the nearer intersection in clockwise direction is given (Q–1), while in O9986 the farther one is specified (Q1).
Circle intersection calculation can also be combined with chamfer or corner rounding specification. E.g.:

```
%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
N70 M30
%
```

In the example a 20mm corner-rounding is programmed in block N30 (,R20). The control calculates the intersection of blocks N30 and N40 and inserts the programmed rounding to the resulting contour.

### 16.3.5 Chaining of Intersection Calculations

**Intersection calculation blocks can be chained**, i.e., more successive blocks can be selected for intersection calculation. The control calculates intersection till straight lines or circles determined over are found.
Let us examine the example below:

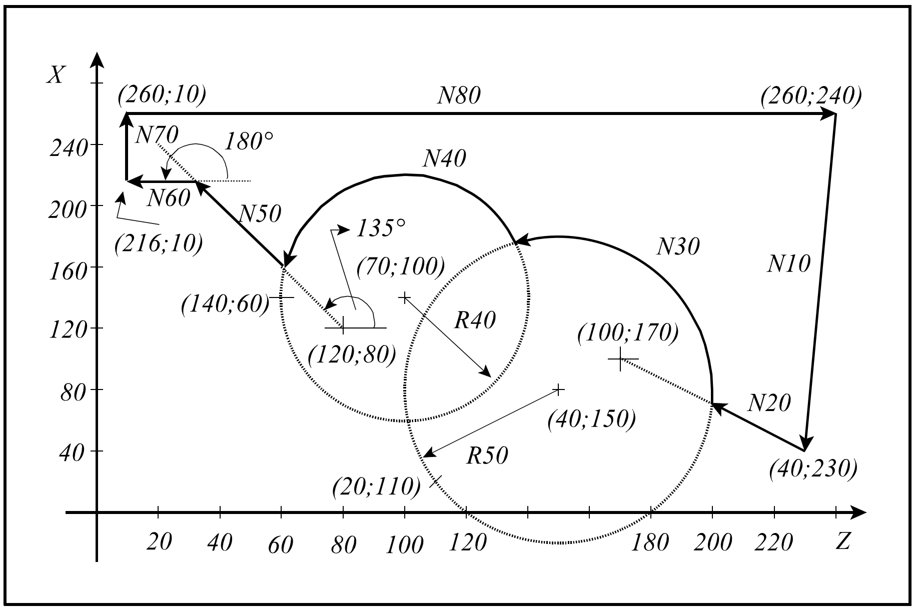


Fig. **16.3.5-1**

```
%O9984
N10 (G18) G0 G42 X40 Z230 F300 S500 M3
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
N90 M30
%
```

In the above example blocks N30, N40, N50, N60 are determined over. Linear block N20 is not drawn to its programmed end point (X100 Z170) because circular block N30 is defined over, i.e., addresses I, J, R are all filled in and the intersection to be searched is given at address Q. Nor circular block N30 is drawn to its programmed end point (X20 Z110) for circular block N40 is also determined over. The last block determined over in the program is the linear block N60. As the following linear block N70 is not defined over, coordinates X216 Z10 programmed in block N60 are not referred to as an intersection point of the straight line but as end point coordinates of block N60.

*In general it is true, that the coordinate points of linear and circular blocks determined over in the selected plane are only referred to by the control as end point coordinates if they are not followed by a block defined over.*

# 17 Canned Cycles for Turning

## 17.1 Single Cycles

The single cycles are the cutting cycle G77, the simple thread cutting cycle G78 and the end face cutting cycle G79.

### 17.1.1 Cutting Cycle (G77)

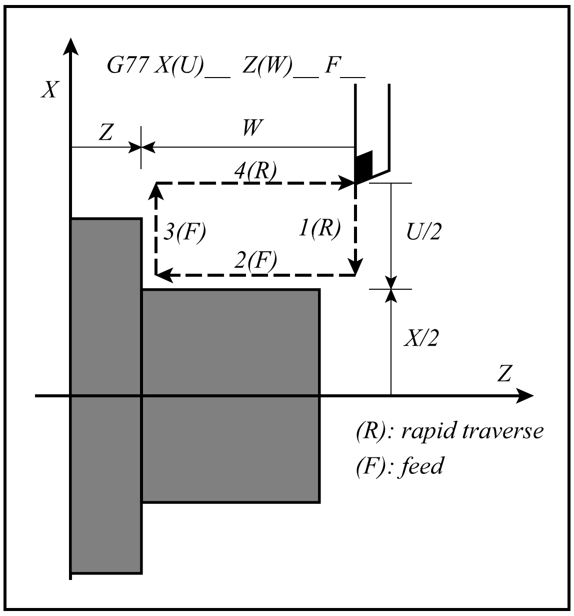


Fig. **17.1.1-1**

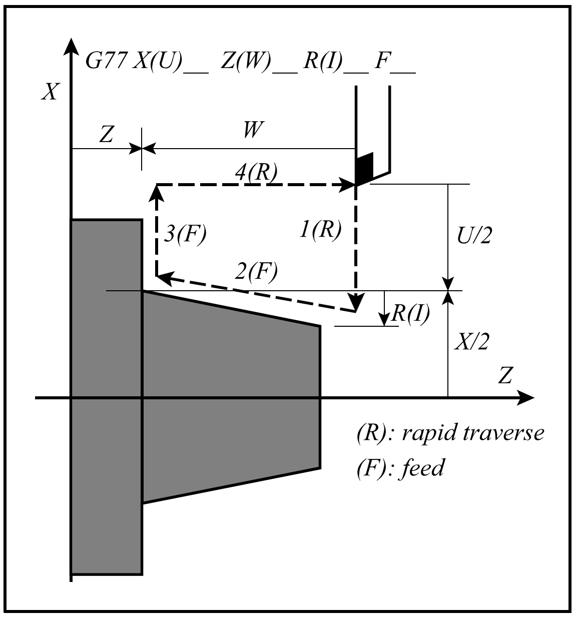


Fig. **17.1.1-2**

**Straight cutting cycle** can be defined in the following way:

**G77** X(U)__ Z(W)__ F__

Incremental programming is also possible with operator I or by programming G91.
In case of incremental programming the data sign determines the direction of pathes Nos. 1 and 2. In the figure the sign of both addresses U and W is negative.
In the block in pathes Nos. 2 and 3 the movement is at feedrate programmed at address F or at the one that is modal, while that in pathes Nos. 1 and 4 is at rapid traverse rate.

**Taper cutting cycle** can be defined in the following way:

**G77** X(U)__ Z(W)__ R(I)__ F__

The taper can be specified at either address R or I. In both cases the data interpretation is the same. The data given at address **R(I)** is **always incremental**, and is acknowledged from the position given at address X(U). The taper slope direction is determined by the sign of address R(I). The interpretation of the other addresses corresponds to those discussed earlier at straight cutting cycle.
Code G77 and data programmed in block G77 are modal.
In single block mode stop is activated at the end of all four actions (1, 2, 3, 4).

In case of incremental programming the signs of addresses U, W and R(I) influence the movement directions as follows:

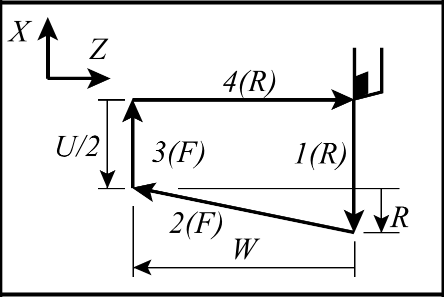
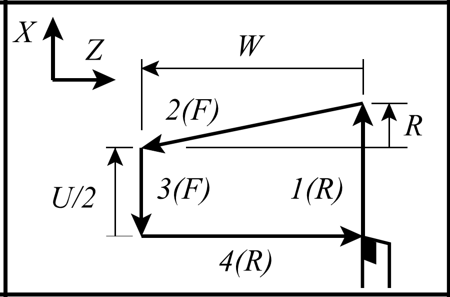
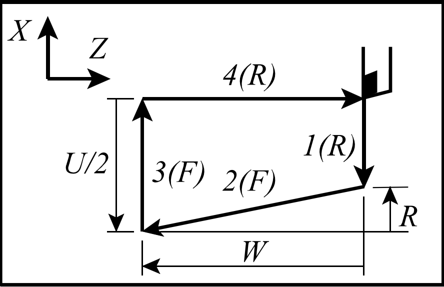
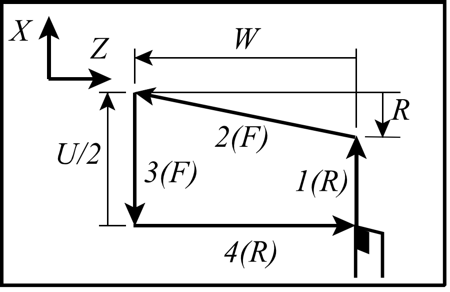
| *1. U<0, W<0, R<0* | *2. U>0, W<0, R>0* |
|---|---|
|  |  |
| *3. U<0, W<0, R>0, és* $\lvert R\rvert \le \lvert U\rvert/2$ | *3. U>0, W<0, R<0, és* $\lvert R\rvert \le \lvert U\rvert/2$ |
|  |  |

Fig. **17.1.1-3**

## 17.1.2 Thread Cutting Cycle (G78)

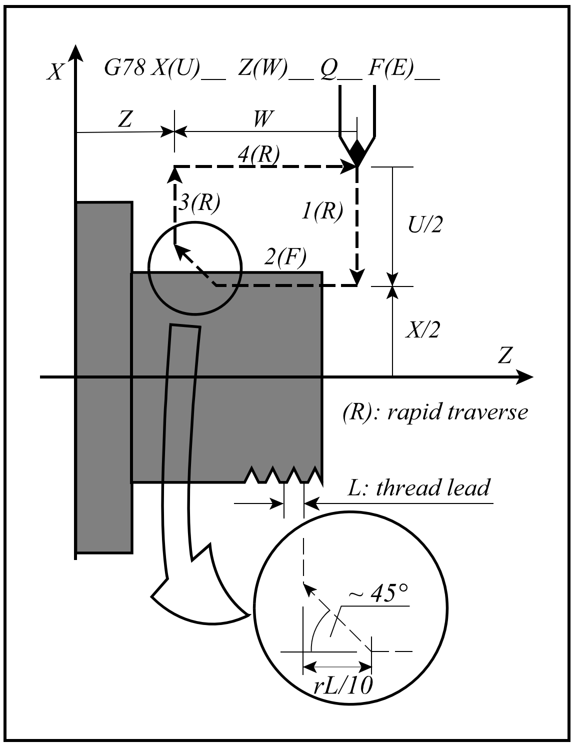


Fig. **17.1.2-1**

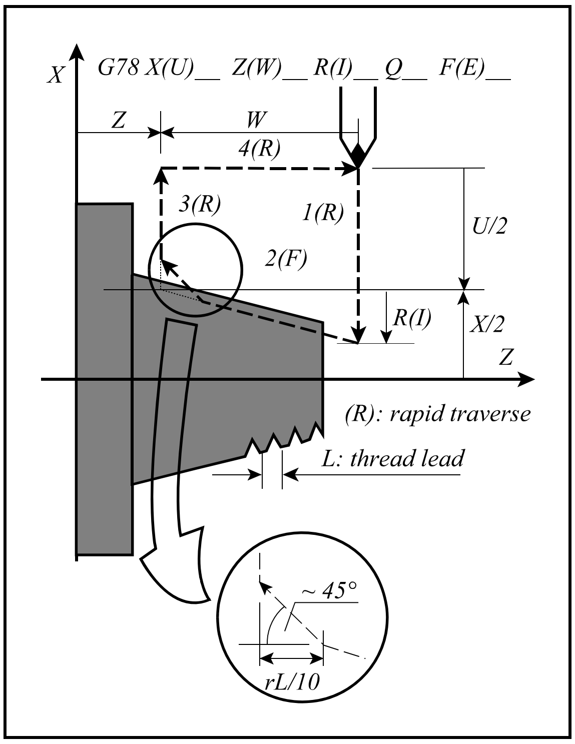


Fig. **17.1.2-2**

**Straight thread cutting cycle** can be defined in the following way:

**G78** X(U)__ Z(W)__ Q__ F(E)__

Incremental programming is also possible with operator I or by programming G91.
In case of programming the data sign determines the direction of pathes Nos. 1 and 2. In the figure the sign of both addresses U and W is negative.
In the block either the thread lead is programmed at address F or the number of ridges pro inch is programmed at address E, as well as the thread start angle in degrees calculated from the encoder zero pulse as written in block G33.
Movements Nos. 1, 3, 4 are at rapid traverse rate.
At the end of path No. 2, where thread cutting occurs, approximately a 45 deg chamfering is executed.
The chamfering amount is determined by parameter 1334 THRDCHMFR indicated with r in the figure. The length of section is

r·L/10

where r: Value of parameter THRDCHMFR
L: Programmed thread lead.

The value of parameter THRDCHMFR ranges from 1 to 255, i.e. the chamfering ranges from 0.1L to 25.5L. If for example the parameter value is 4 and the programmed thread lead is F2 then the chamfering amount:

2*(4/10)=0.8mm

**Taper thread cutting cycle** can be defined in the following way:

**G78** X(U)__ Z(W)__ R(I)__ Q__ F(E)__

The taper can be specified at either address R or I. In both cases the data interpretation is the same. The data given at address **R(I)** is **always incremental**, and is acknowledged from the position given at address X(U). The taper slope direction is determined by the sign of address R(I). The interpretation of the other addresses corresponds to those discussed for straight thread cutting cycle.
The chamfering angle is in this case also 45° and chamfering amount r is measured along the straight line parallel to the axes.
Code G78 and data programmed in block G78 are modal.
In single block mode stop is activated at the end of all four actions (1, 2, 3, 4).

## Effect of STOP button in cycle action No. 2

Actions of cycles 1, 3 and 4 can be stopped by the use of STOP button any time and the slides stop as in case of normal interpolation G0.
The press of STOP button is also effective in thread cutting phase No. 2, however in this case the control first executes the same chamfering as at the end of action No. 2, afterwards it first retracts along X axis at rapid traverse rate, than moves to start point in Z. STOP button is not effective in the escape path.

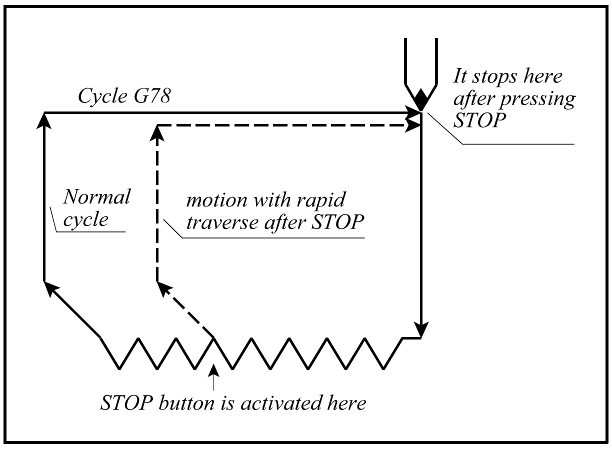

It stops here after pressing STOP
Cycle G78
Normal cycle
motion with rapid traverse after STOP
STOP button is activated here

Fig. **17.1.2 -3**

### 17.1.3 End Face Cutting Cycle (G79)

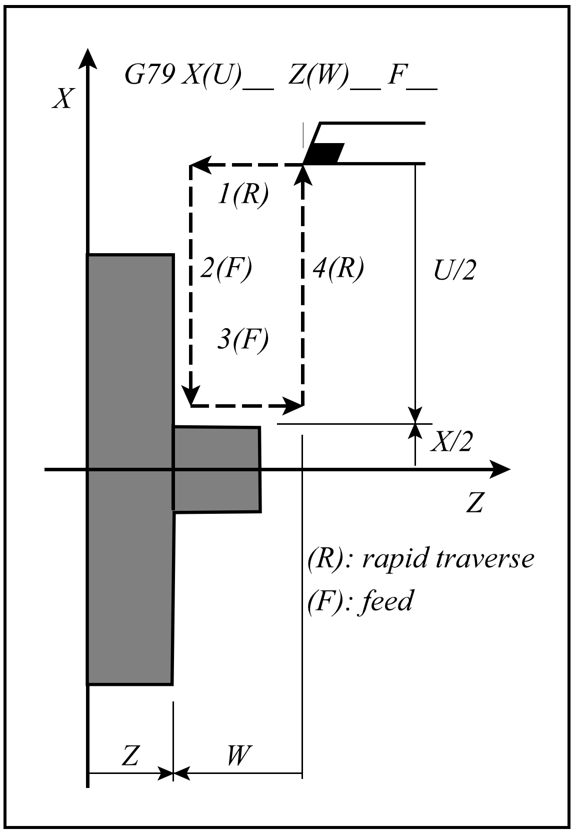


Fig. 17.1.3 -1

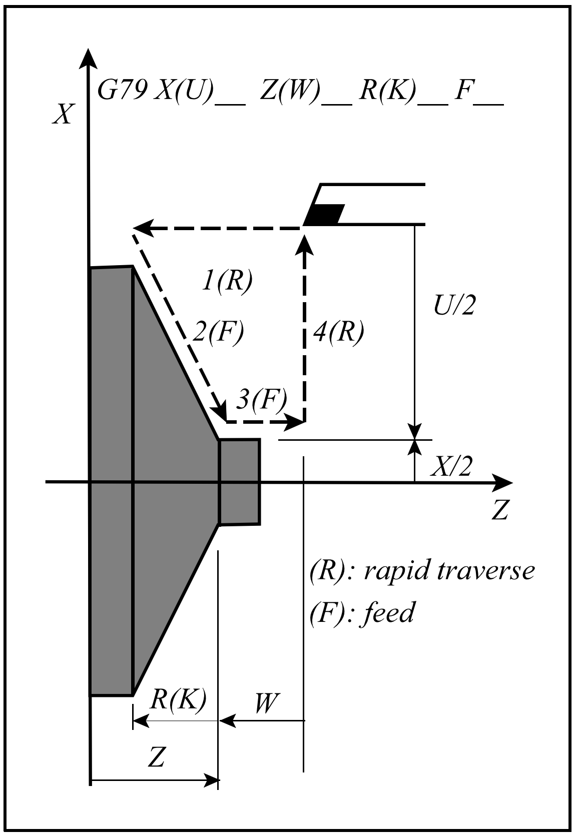


Fig. 17.1.3 -2

**Straight face cutting cycle** can be defined in the following way:

**G79** X(U)__ Z(W)__ F__

Incremental programming is also possible with operator I or by programming G91.
In case of incremental programming the data sign determines the direction of pathes Nos. 1 and 2. In the figure the sign of both addresses U and W is negative.
In the block in pathes Nos. 2 and 3 the movement is at feedrate programmed at address F or modal, while that in pathes Nos. 1 and 4 is at rapid traverse rate.

**Taper face cutting cycle** can be defined in the following way:

**G77** X(U)__ Z(W)__ R(K)__ F__

The taper can be specified at either address R or K. In both cases the data interpretation is the same. The data given at address **R(K)** is **always incremental**, and is acknowledged from the position given at address X(U). The taper slope direction is determined by the sign of address R(K). The interpretation of the other addresses corresponds to those discussed for straight face cutting cycle.
Code G79 and data programmed in block G79 are modal.
In single block mode stop is activated at the end of all four actions (1, 2, 3, 4).

In case of incremental programming the signs of addresses U, W and R(K) influence the movement directions as follows:

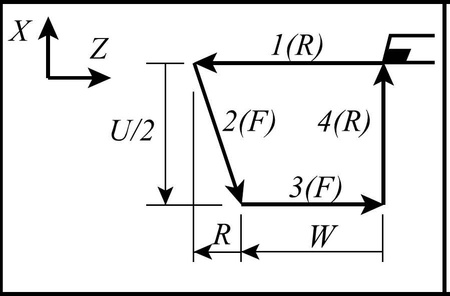
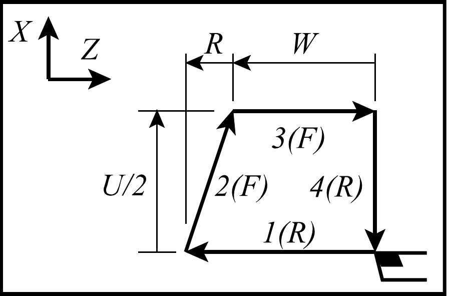
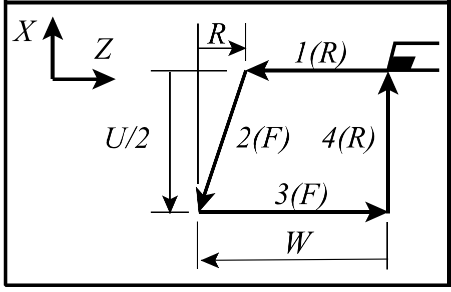
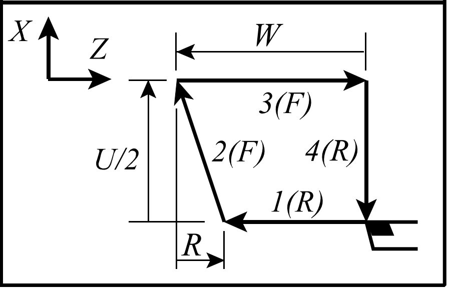
| *1. U<0, W<0, R<0* | *2. U>0, W<0, R<0* |
|---|---|
|  |  |
| *3. U<0, W<0, R>0, és \|R\|≤\|W\|* | *3. U>0, W<0, R>0, és \|R\|≤\|W\|* |
|  |  |

Fig. **17.1.4 -3**

## 17.1.4 Single Cycle Application

Both G codes and input parameters of cycles are modal. This means that if the cycle variables X(U), Z(W) or R(I or K), are already given and their values have not changed, they must not be rewritten in pogram.. For example:

```
        G91...
        G77 X–20 Z–50 F0.5
        X–30
        X–40
        X–50
        ...
```

In the above example only the value of depth of cut (X) changes, therefore only this address must be refilled, the other values remain unchanged.

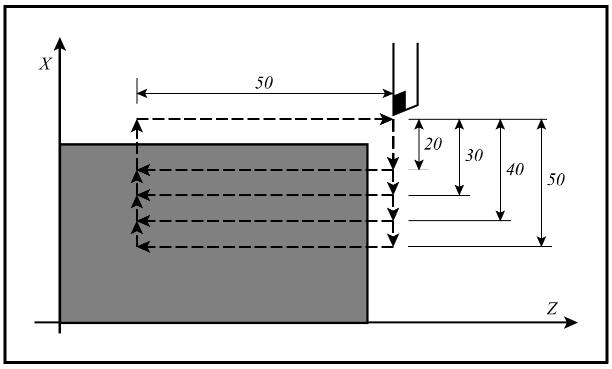


Fig. **17.1.4 -1**

The cycle is executed in the switched-on state of the cycle provided a variable of X(U), Z(W) or R(I or K) referring to movement is also filled in. If for example in cycle state, function is executed in a separate block, the cycle state remains switched-on but the cycle is not repeated:

```
        ...
        G77 U–20 W–50 F0.5       (switch on and execute cycle)
        T202                     (cycle switched on but not executed)
        U–30                     (cycle executed)
        ...
```

The cycle and the modal variables are deleted by interpolation G codes belonging to group No. 1 as well as by all single-shot G codes except for dwell G4.

M, S, T function can also be written in blocks containing simple cycles. The functions are always executed in action No. 1 of the cycle either parallel to movement or at the end of it. If in certain cases this may cause an inconvenience, the function shall be written in a separate block.

## 17.2 Multiple Repetitive Cycles

Multiple repetitive cycles simplify the writing of machining programs. For example the profile of the part must be specified for finishing. At the same time this profile determines the basis of cycles executing stock removal (G71, G72, G73). Other than the stock removal also a finishing cycle (G70), a thread cutting cycle (G76) and two drilling cycles (G74, G75) are available.

### 17.2.1 Stock Removal in Turning (G71)

There are two kinds of stock removals in turning: Type 1 and 2.

**Stock removal in turning type 1**

If a finished shape of A–A'–B is given by the program as in the figure, the specified area is removed with Δd depth of cuts with finishing allowance Δu/2 and Δw left.

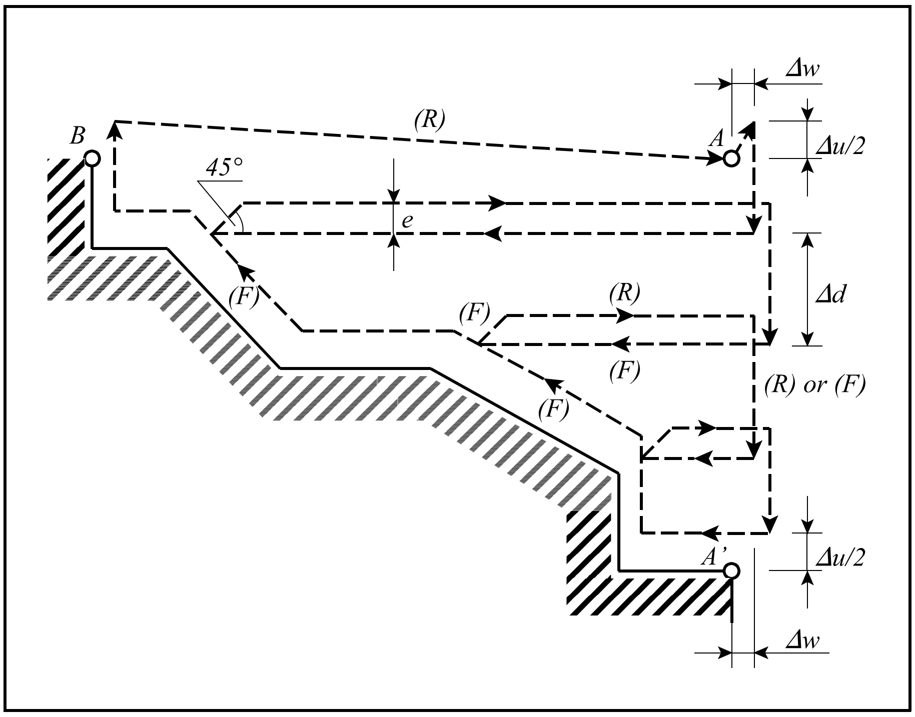


Fig. **17.2.1 -1**

**1st specification method:**
With command lines:

```
          G71  U(Δd)  R (e)
          G71  P (ns)  Q (nf)  U(Δu)  W(Δw)  F(f)  S(s)  T(t)
                   N(ns)  X(U) ...
                   ...
                            F___
                            S___
                            T___
                   N(nf)  ...
```

where:

**Δd**: Depth of cut. **Positive** number interpreted always in **radius**. The depth of cut can also be given at parameter 1339 DPTHCUT as well as this parameter is overwritten as the effect of the program command. This also means that in case the depth of cut is not specified the control refers to this parameter.

**e**: Escaping amount. **Positive** number interpreted always in **radius**. The escaping amount can also be given at parameter 1340 ESCAPE as well as this parameter is overwritten as the effect of the program command. This also means that in case the escaping amount is not specified the control refers to this parameter.

**$n_s$**: Sequence number of the first block for the program of finishing shape (phase A–A'–B).

**$n_f$**: Sequence number of the last block for the program of finishing shape (phase A–A'–B).

**Δu**: Distance and direction of finishing allowance along X axis. **Signed** number interpreted in **diameter** or **radius** in function of X coordinate.

**Δw**: Distance and direction of finishing allowance along Z axis. **Signed** number.

**f**, **s**, **t**: Any F, S or T function programmed in blocks $n_s$ to $n_f$ in the program for finishing shape (phase A–A'–B) is ignored and values f, s, t in block G71 are effective during the cycle.

The meaning of the value specified at address U being Δd or Δu depends on wether P and Q were programmed in the given block. If not the meaning of address U is Δd, if yes it is Δu.
The stock removal is executed by the block in which P and Q had been specified.
The movement between points **A–A'** must be specified in block No. $n_s$ given at address P **obligatorily** by programming **G00** or **G01**. The code defined here decides if the **cutting in** (movement in direction A–A') is done at rapid traverse rate (in case programming **G00**) or at feedrate (in case programming **G01**) in the course of stock removal. In this block **no** movement in direction P($n_s$) **Z** shall be given.
Phase A'–B is the profile made up of straight lines and arcs. The profile must be monotonic increasing or decreasing both in X and Z directions which means that regression is not possible in either way. The cycle can be used in all four quadrants. The figure also shows the sign of finishing allowance.
F, S, T functions programmed in blocks $n_s$ to $n_f$ are omitted and those programmed in block G71 (f, s, t) or previously are effective. The same is true for constant surface speed programmed in blocks $n_s$ to $n_f$, i.e., state G96 or G97 and value of surface speed preceding block G71 is effective.

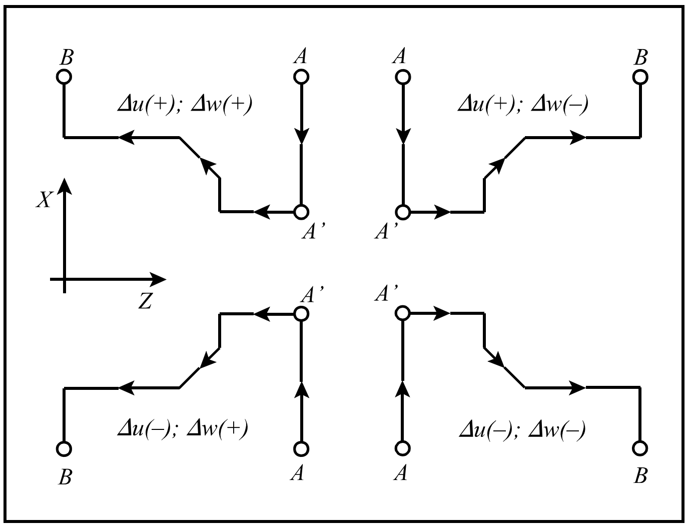


Fig. 17.2.1 -2

Blocks $n_s$ to $n_f$ cannot contain subprogram calls.
Tool nose radius compensation calculation (G41, G42) can be switched on during cycle execution with the obligation that it must be switched on (G41 or G42) and off (G40) between blocks ranging from $n_s$ to $n_f$:

**CORRECT**

```
N(ns) X(U) G41 ...
      (G41)...
       ...
      (G40)
N(nf)  G40 ...
```

**or**

```
       G41
N(ns)  X(U) ...
      ...
      ...
N(nf)  ...
      G40
```

**INCORRECT**

```
     G41
N(ns) X(U) ...
     ...
     ...
     G40
N(nf) ...
```

**or**

```
N(ns) G41 X(U) ...
      ...
      ...
N(nf)  ...
      G40
```

If the cycle is interrupted during execution, edit mode is opened and parameter 1339 DPTHCUT is overwritten, than in automatic mode the program is run by means of START the next cut is already taken into account with the new depth of cut. The same goes for parameter 1340 ESCAPE i.e., for escaping amount.

**2nd specification method:**

**G71** **P** ($n_s$) **Q** ($n_f$) **U**(Δu) **W**(Δw) **D**(Δd) **F**(f) **S**(s) **T**(t)
N($n_s$) X(U) ...
...
F___
S___
T___
N($n_f$) ...

Input parameters of the second specification method correspond to that of the first one.

### Stock removal in turning type 2

Stock removal in turning type 2 must be defined like type 1, its code is G71 and its input parameters are also the same as those of type 1. The difference is the specification of the start up block of the finishing shape (block No. $n_s$). While in case of calling type 1 address Z cannot be referred to in this block, i.e., the movement of phase A–A' must be perpendicular to Z axis, in case of calling type 2 address Z must be obligatorily referred to in this block. In other words phase A–A' must not be perpendicular to Z axis.

**Specification of type 1**

```
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
N100 X(U)___
...
...
...
N200
```

**Specification of type 2**

```
G71 U8 R1
G71 P100 Q200 U0.5 W0.2
N100 X(U)___ Z(W)__
...
...
...
N200
```

In case type 2 must be used with movement only in X direction, i.e., perpendicular to Z axis in the first block of finishing shape, incremental movement 0, i.e. ZI0 or W0 must be programmed along Z axis.
Stock removal in turning type 2 differs from type 1 in the profile not necessary being monotonic increasing or decreasing in X direction, i.e., the profile may have concaves. The cycle can handle at most 10 reflex pockets.

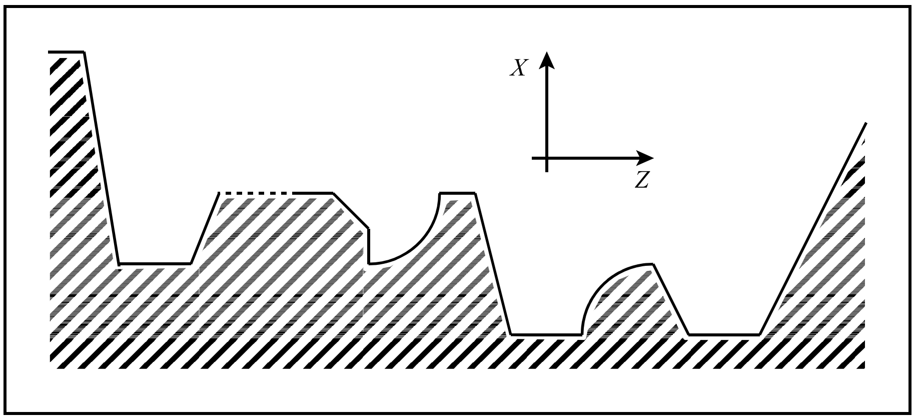


Fig. **17.2.1 -3**

On the other hand **in Z direction** the profile must remain **monotonic,** it cannot include reflex.

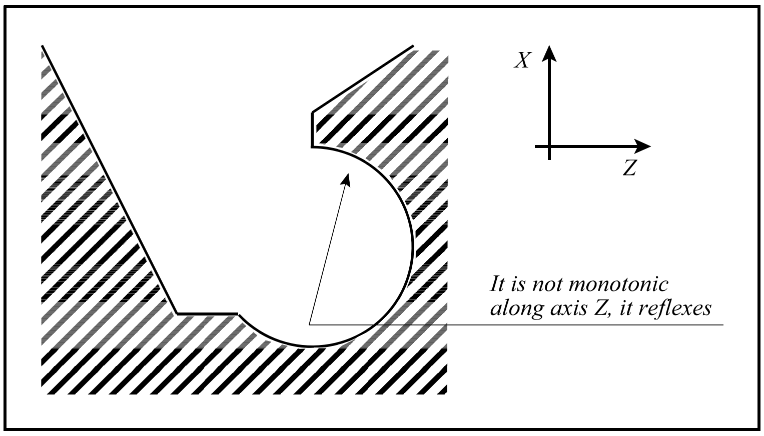


Fig. **17.2.1 -4**

The first block of the finishing shape ($n_s$) may also contain Z-direction movement (moreover address Z must be obligatorily referred to), i.e., the first cut must not be perpendicular to Z axis.

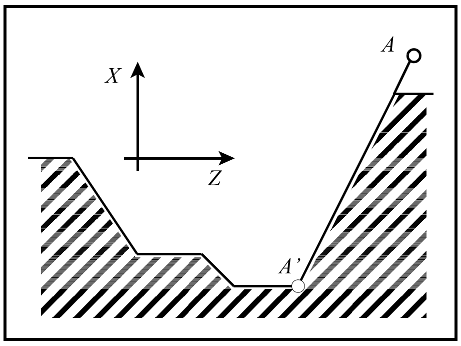


Fig. **17.2.1 -5**

In case of stock removal in turning type 2 the escaping amount is perpendicular to axis Z and is done with valid escape amount “e”.

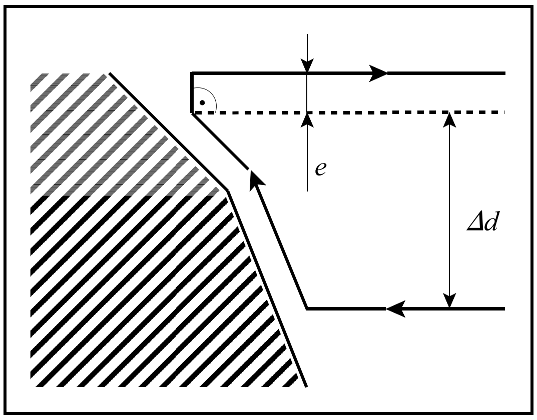


Fig. **17.2.1 -6**

The below figure shows an example how does the cycle cut the rough workpiece.

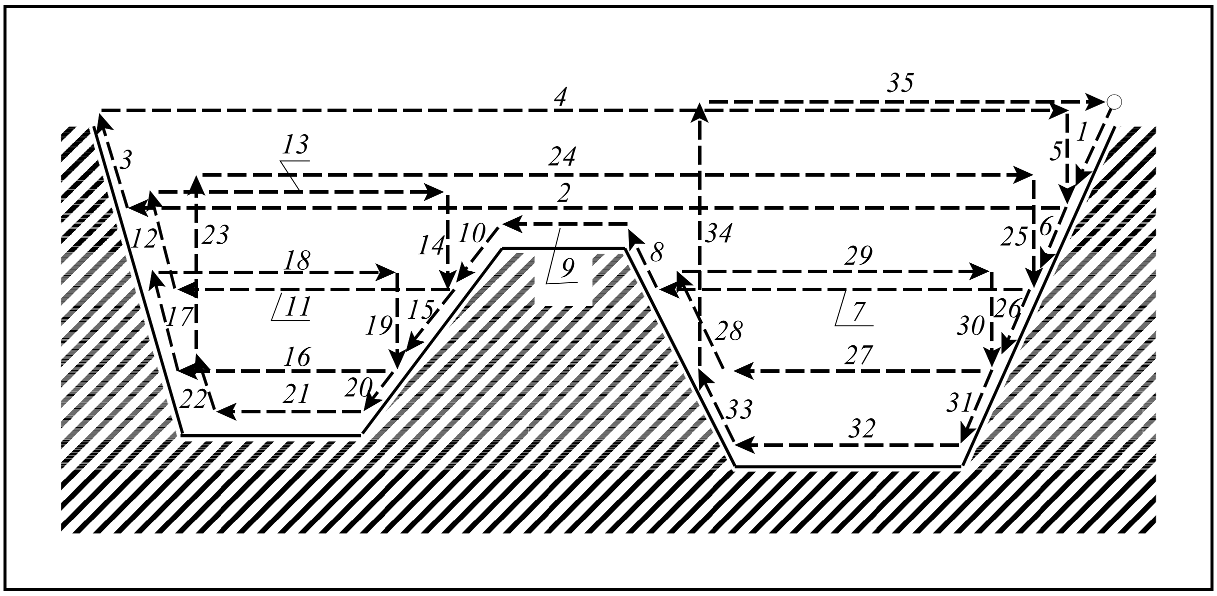


Fig. **17.2.1 -7**

In the above case Z-direction finishing allowance (Δw) cannot be programmed (it must be W0), otherwise the tool may cut in a side wall.

## 17.2.2 Stock Removal in Facing (G72)

There are two kinds of stock removals in facing: Type 1 and 2.

**Stock removal in facing type 1**

The stock removal in facing (G72) seen in the below figure is the same as stock removal in turning G71 except for that cutting is made by an operation parallel to X axis.

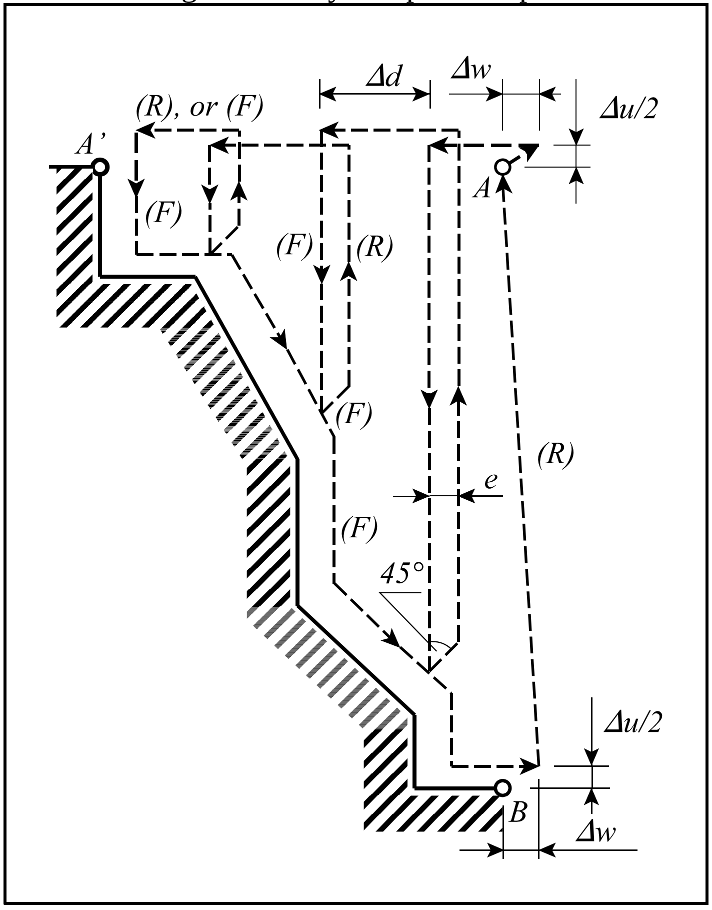


Fig. 17.2.2 -1

**1st specification method:**

**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) ...
...
F___
S___
T___
N($n_f$) ...

The meaning of input parameters thoroughly correspond to those discussed for cycle G71.

**2nd specification method:**

```
G72 P(n_s) Q(n_f) U(Δu) W(Δw) D(Δd) F(f) S(s) T(t)
        N(n_s) Z(W) ...
        ...
                F___
                S___
                T___
        N(n_f) ...
```

The cycle can be used in all four quadrants. The figure shows the finishing allowance sign for all four cases.
In block No. $n_s$ describing movement between points A–A' reference to X axis is not possible, the movement must always be parallel to Z axis. The interpolation code (G00 or G01) defined in block No. $n_s$ determines if the cut in is done at feedrate or rapid traverse rate.
The programmed profile must be monotonic, i.e., continuously increasing or decreasing along both axes.
For tool radius compensation see cycle G71.

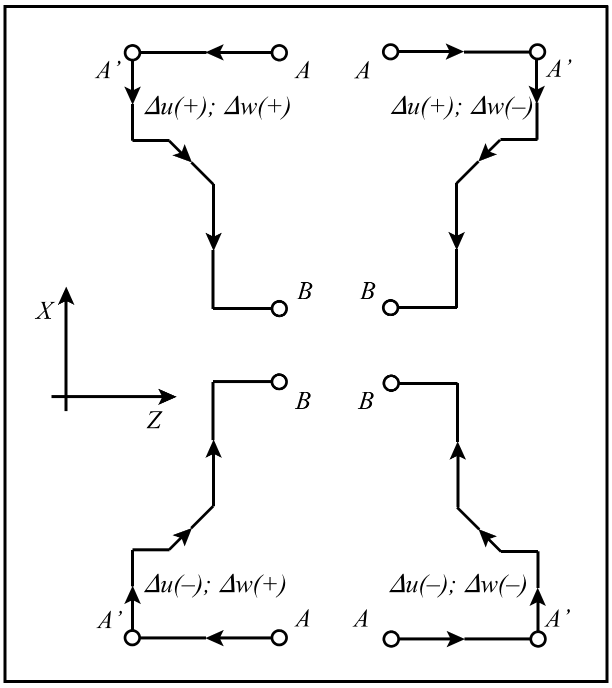


Fig. 17.2.2 -2

### Stock removal in facing type 2

Stock removal in facing type 2 must be defined like type 1, its code is G72 and its input parameters are also the same as those of type 1. The difference is the specification of the start- up block of the finishing shape (block No. $n_s$). While in case of calling type 1 address X cannot be referred to in this block, i.e., the movement of phase A–A' must be perpendicular to X axis, in case of calling type 2 address X must be obligatorily referred to in this block. In other words phase A–A' must not be perpendicular to X axis. Further on the same obligations are valid for the cycle as for G71 type 2. The profile can also be reflex, however it must be monotonic in X direction.

## 17.2.3 Pattern Repeating Cycle (G73)

This cycle can be used for cutting parts whose rough shape has already been made by rough machining, forging or casting method. This function permits cutting a fixed pattern repeatedly, with a pattern being displaced bit by bit.

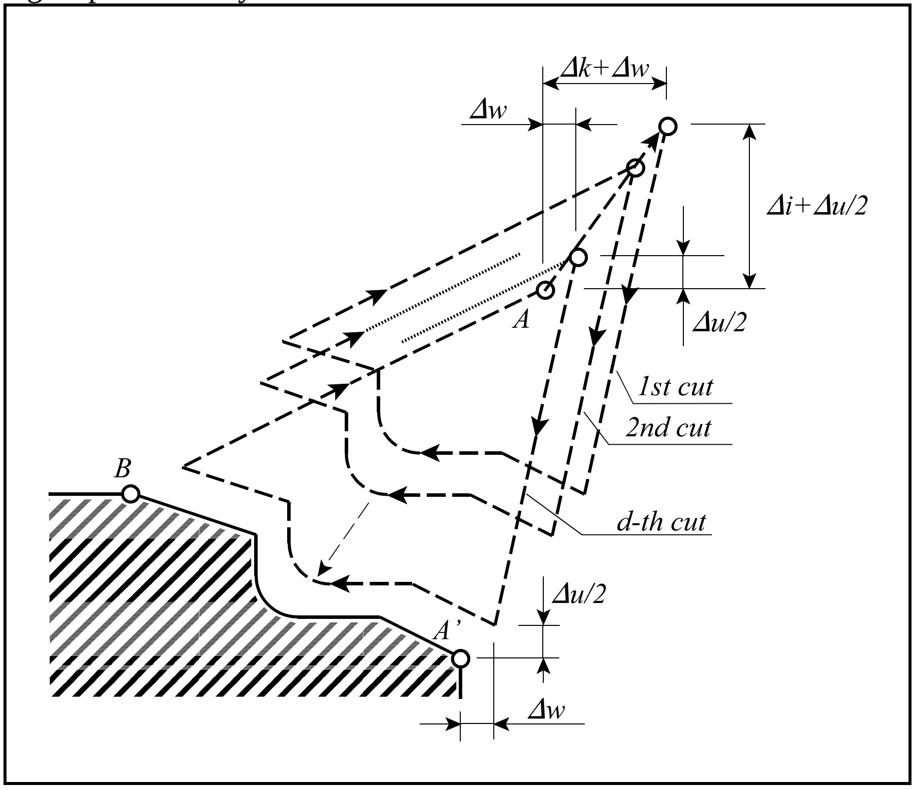


Fig. **17.2.3 -1**

**1st specification method:**
Command

```
        G73  U(Δi)  W(Δk)  R (d)
        G73  P (ns)  Q (nf)  U(Δu)  W(Δw)  F(f)  S(s)  T(t)
                N(ns) ...
                ...
                        F___
                        S___
                        T___
                N(nf) ...
```

where:

**Δi**: Distance and direction of relief along X axis. **Signed** number always interpreted in **radius**. The relief can also be given at parameter 1341 RELIEFX as well as this parameter is overwritten as the effect of program command.

**Δk**: Distance and direction of relief along Z axis. **Signed** number always interpreted in **radius**. The relief can also be given at parameter 1342 RELIEFZ as well as this parameter is overwritten as the effect of program command

**d**: Number of division. The number of division can also be given at parameter 1343 NUMDIV as well as this parameter is overwritten as the effect of program command. The value specified for relief (parameters RELIEFX, RELIEFZ) is divided by this number and the advances are taken with the result in the course of stock removal.

**$n_s$**: Sequence number of the fist block for the program of finishing shape (phase A–A'–B).

**$n_f$**: Sequence number of the last block for the program of finishing shape (phase A–A'–B).

**Δu**: Distance and direction of finishing allowance along X axis. **Signed** number interpretated in **diameter** or **radius** in function of X coordinate.

**Δw**: Distance and direction of finishing allowance along Z axis. **Signed** number.

The meaning of addresses U and W defined in block G73 being Δi and Δk or Δu and Δw depends on if P and Q have been programmed in the given block. That is if P and Q have been programmed, then the meaning of U and W is Δu and Δw, if not, then it is Δi and Δk.
The cycle is executed in the block containing P and Q. Blocks from $n_s$ to $n_f$ must contain the positioning (phase A–A') as well as the description of the finishing shape (phase A'–B). The cycle can be executed in all four quadrants in function of the sign of values Δi, Δk, Δu, Δw. At the end of cycle the tool returns to point "A". The machining is continued from the following block.
F, S, T functions programmed in blocks $n_s$ to $n_f$ are omitted and those programmed in block G71 (f, s, t) or previously are effective. The same is true for constant surface speed programmed in blocks $n_s$ to $n_{f,}$, i.e. state G96 or G97 and constant surface speed value preceding block G71 is effective.
Blocks ranging from $n_s$ to $n_f$ cannot contain subprogram call.
Tool nose radius compensation specification is possible in blocks defining the finishing shape with the restrictions discussed in description G71.

**2nd specification method:**

**G73 P** ($n_s$) **Q** ($n_f$) **U**(Δu) **W**(Δw) **I**(Δi) **K**(Δk) **D**(d) **F**(f) **S**(s) **T**(t)

N($n_s$) ...

...

F___

S___

T___

N($n_f$) ...

Input parameters of the second specification method correspond to those of the first one.

## 17.2.4 Finishing Cycle (G70)

After the stock removal with G71, G72 or G73 finishing can be defined by means of command G70. Finishing can be given with the following command:

**G70 P** ($n_s$) **Q** ($n_f$) **U**(Δu) **W**(Δw)

**$n_s$**: Sequence number of the first block for the program of finishing shape.

**$n_f$**: Sequence number of the last block for the program of finishing shape.

**Δu**: Distance and direction of finishing allowance along X axis.. **Signed** number interpreted in **diameter** or **radius** in function of X coordinate.

**Δw**: Distance and direction of finishing allowance along Z axis. **Signed** number.

**F, S, T** functions programmed in the program part between blocks $n_s$ and $n_f$ are **executed** during the cycle contrary to cycles G71, G72, G73.
At the end of finishing cycle the tool returns to start point and the following block is loaded.
Tool **radius compensation** calculation **works** in the course of finishing cycle.
Finishing allowance can be defined at addresses U and W in case the finishing allowance is to be removed in more steps.
Blocks ranging from $n_s$ to $n_f$ cannot contain subprogram call.

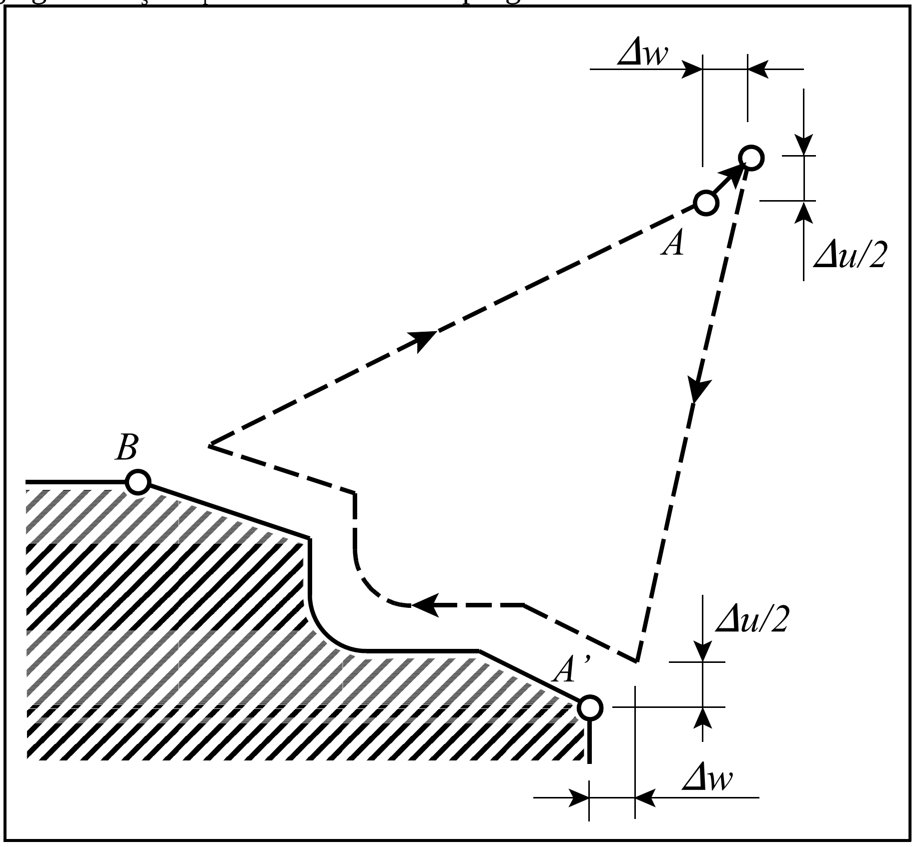


Fig. **17.2.4-1**

## 17.2.5 End Face Peck Drilling Cycle (G74)

The enclosed figure shows the process of end face peck drilling cycle G74. The drilling is in Z direction.

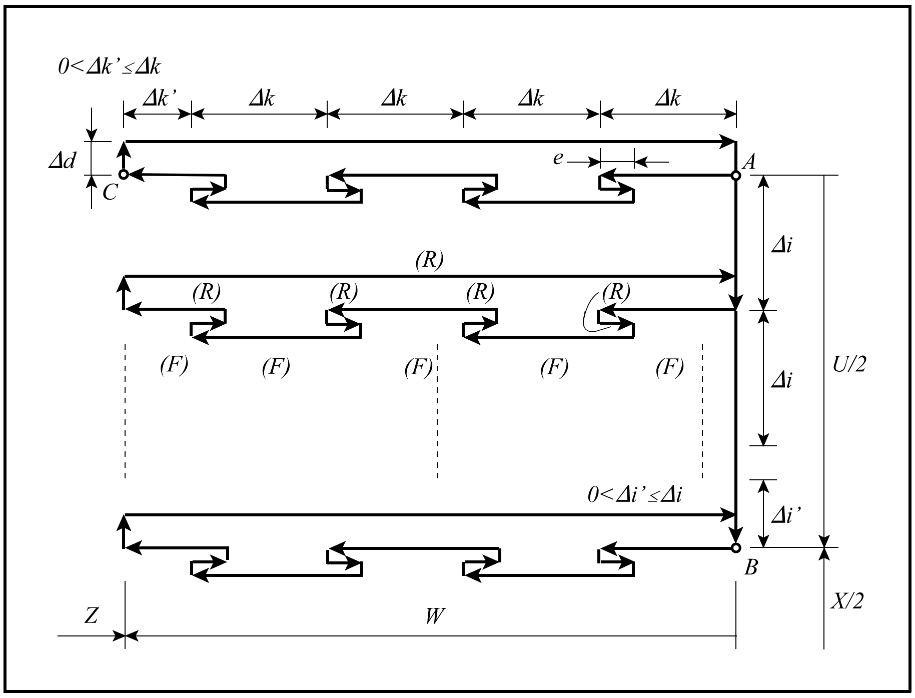


Fig. 17.2.5 -1

**1st specification method:**

Command line

**G74 R** (e)

**G74 X**(**U**) **Z**(**W**) **P** (Δi) **Q** (Δk) **R** (Δd) **F**

where:

**e**: Return amount
Modal value, unchanging until overwritten. The return amount can also be given at parameter 1344 RETG74G75, as well as this parameter is overwritten as the effect of program command.

**X**: X component of point B

**U**: Incremental amount from A to B

**Z**: Z component of point C

**W**: Incremental amount from A to C

**Δi**: Movement amount in X direction. The number is positive and always interpreted in **radius.**

**Δk**: Depth of cut in Z direction. The number is always positive

**Δd**: Relief amount at the cutting bottom. The sign of Δd is always positive (the movement direction is always contrary to vector sign AB).

However, if the filling out of addresses X(U) and P(Δi) is omitted the sign of R(Δd) is interpreted and the movement direction is determined by the sign of Δd at the cutting bottom.

**F**: Feedrate

In the figure (F) indicates the phases done at feedrate and (R) indicates those done at rapid traverse rate.
The filling out of address R in block G74 determining e or Δd depends upon the filling out of address Z(W). If address Z(W) is filled out the interpretation of address R is Δd.
If the filling out of both addresses X(U) and P(Δi) is omitted movement occurs only along Z axis, i.e. drilling cycle takes place.

**2nd specification method:**

**G74 X**(**U**) **Z**(**W**) **I** (Δi) **K** (Δk) **D** (Δd) **F**

The input parameters of the second specification method correspond to those of the first one.

## 17.2.6 Outer Diameter/Internal Diameter Drilling Cycle (G75)

The enclosed figure shows the process of outer diameter/internal diameter drilling cycle G75.

**1st specification method:**

**G75 R** (e)
**G75 X**(**U**) **Z**(**W**) **P** (Δi) **Q** (Δk) **R** (Δd) **F**

The interpretation of cycle variables corresponds to that of cycle G74 with the difference that the drilling is in X direction, therefore the interpretation of addresses X(U) and Z(W) is reversed.

**2nd specification method:**

**G75 X**(**U**) **Z**(**W**) **I** (Δi) **K** (Δk) **D** (Δd) **F**

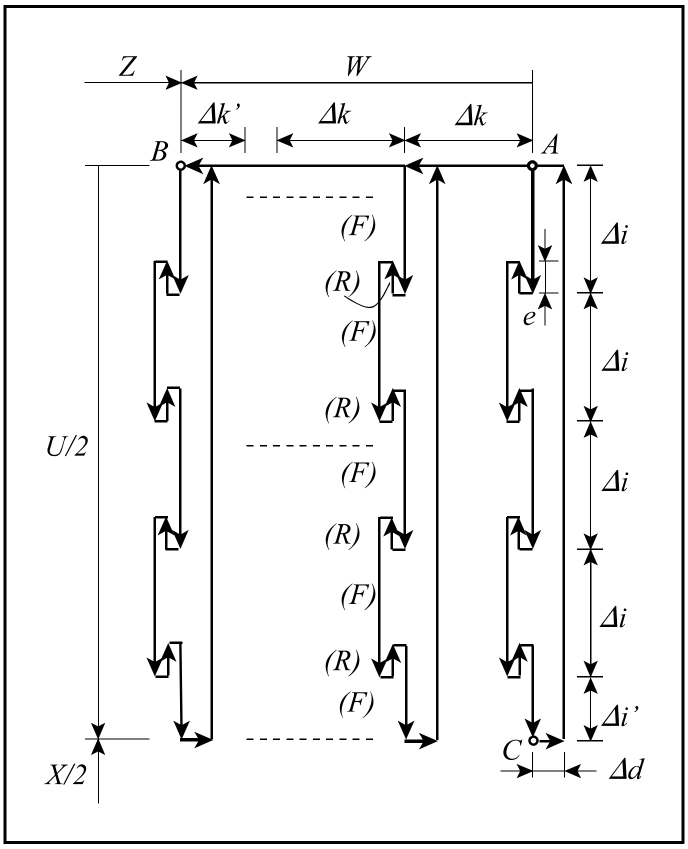


Fig. 17.2.6 -1

## 17.2.7 Multiple Thread Cutting Cycle (G76)

The enclosed figure shows the process of multiple thread cutting cycle G76.

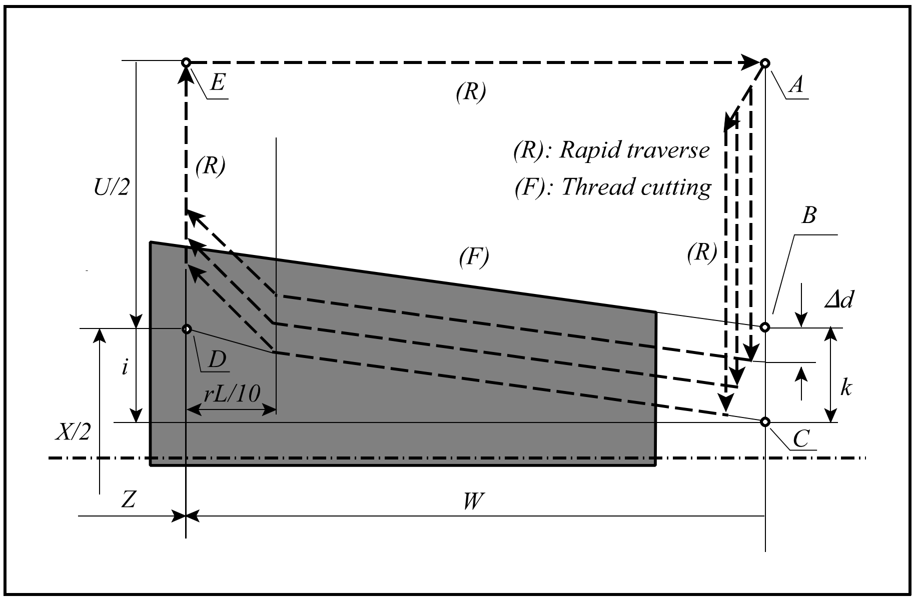


Fig. **17.2.7 -1**

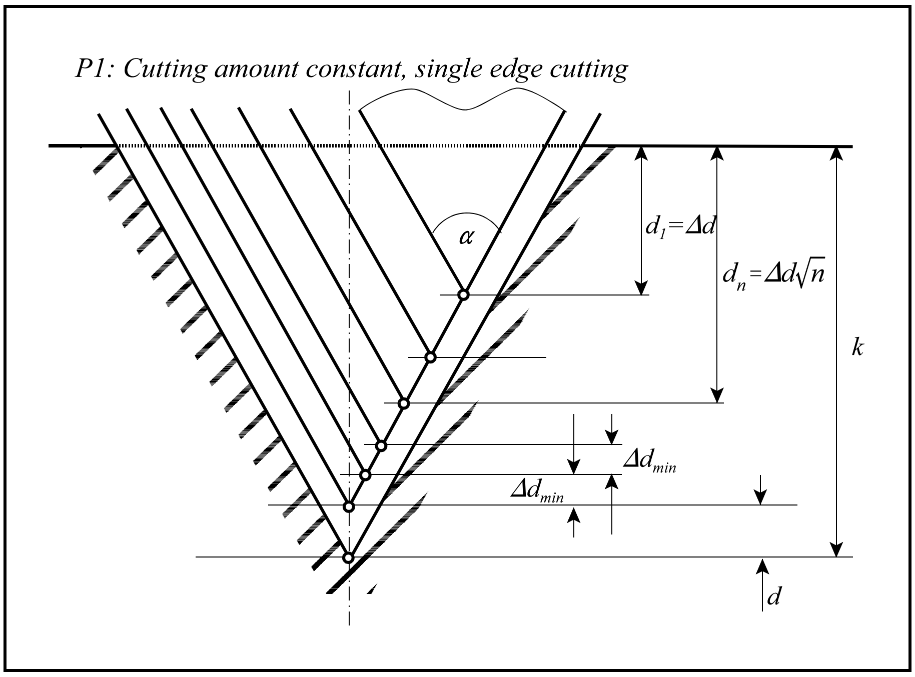


Fig. **17.2.7 -2**

**1st specification method:**

Command lines

**G76 P** (n) (r) ($\alpha$) **Q** ($\Delta d_{min}$) **R** (d)

**G76 X**(**U**) **Z**(**W**) **P** (k) **Q** ($\Delta d$) **R** (i) **F**(**E**)(L)

where:

**n**: **Repetitive count in finishing** (n=01...99)
The value is modal, unchanging until overwritten. The repetitive count of finishing can also be given at parameter 1335 COUNTFIN as well as this parameter is overwritten as the effect of program command.

**r**: **Chamfering amount** (r=01...99)
the tool is escaped in 45°during run out from thread. The chamfering amount can be given with the help of r. The phase length is

$r \cdot L/10$

where: L: The programmed thread lead.
This value is modal, unchanging until overwritten. The chamfering amount can also be given at parameter 1334 THRDCHMFR as well as this parameter is overwritten as the effect of program command.

**$\alpha$**: **Angle of tool tip** in degree ($\alpha$=01...99)
This value is modal, unchanging until overwritten. The angle of tool tip can also be given at parameter 1336 TIPANGL as well as this parameter is overwritten as the effect of program command.

Values n, r and $\alpha$ can be specified altogether at address P. As all values are figured with two-digit numbers, a six-digit number must be written at address P. For example if the number of finishing cycles is n=2, the chamfering amount is 1.5L (r=15) and a tool having a tip of 60° is applied, then value P: **P021560**.

**$\Delta d_{min}$**: **Minimum cutting depth** (positive number interpreted always in **radius**)
If in course of thread cutting in cycle No. n the cutting depth is $d_n - d_{n-1} < \Delta d_{min}$, then the cutting depth is always clamped to value $\Delta d_{min}$. This value is modal, unchanging until overwritten. The minimum cutting depth value can also be given at parameter 1337 MINTHRDP as well as this parameter is overwritten as the effect of program command.

**d**: **Finishing allowance** (positive number interpreted always in **radius**)
Ths value is modal, unchanging until overwritten. The finishing allowance value can also be given at parameter 1338 FINALLW as well as this parameter is overwritten as the effect of program command.

The above listed parameters are the data of first block **G76 P** (n) (r) ($\alpha$) **Q** ($\Delta d_{min}$) **R** (d). The control executes the receiving of the above parameters as the effect of code G76 provided neither address X(U) nor Z(W) is filled out in block G76.

**i**: **Taper size** (interpreted always in **radius**)
If i=0 or address R is not filled out ordinary linear thread cutting can be made.

**k**: **Height of thread** (positive number interpreted always in **radius**)

**$\Delta d$**: **Depth of cut in 1st cut** (positive number interpreted always in **radius**)

**L**: **Lead of thread**
Its programming corresponds to that of G33. Value written at address F indicates thread lead, while the value written at address E indicates the ridges pro inch.

The above parameters are the input data of second block **G76 X**(**U**) **Z**(**W**) **R** (i) **P** (k) **Q** (Δd) **F**(**E**)(L). Thread cutting is only executed as the effect of the above block, which means that one of addresses X(U), Z(W) must be filled out. If neither coordinate is filled out the block is interpreted as parameter setting block.
In the course of cycle execution the **cut in**, that is the movement between points A and C is done at **rapid traverse rate**, provided code **G00** is in effect during cycle, while it is done at modal **feed**, provided code **G01** is in effect during cycle.
Feed movement is executed between points C and D in accordance with thread lead L specified at address F(E). All other phases are executed at rapid traverse rate.
The thread is always cut on one side on the basis of relation $d_n = \Delta d\sqrt{n}$ for the nth path with the cutting amount per cycle always held constant..
Chamfering is always done at the end of thread in compliance with the parameters set.

| | |
|---|---|
| **X(U)**, **Z(W)** | the sign of these addresses defines the direction of cut in and thread cutting |
| **R(i)** | the sign of this address defines the taper slope direction. |

As the effect of STOP the tool is escaped in accordance with the programmed chamfering as discussed in case of code G78 than the start point (A) is positioned. With START button pressed the interrupted cut is started from the beginning.

**2nd specification method:**

**G76 X(U) Z(W) I**(i) **K**(k) **D**(Δd) **A**(α) **F**(**E**)(L) **Q P**

Interpretation of data i, k, Δd, L corresponds to those discussed for the 1st method.
Similarly to the 1st method the angle of tool tip can be defined at address “A” (α). The difference is in the unit specifiable being 1° for the 1st method, while 0.001° for the 2nd one. In case address “A” is not filled out value α is taken from parameter. However the specification of address “A” does not change the value of parameter 1336 TIPANGL.
Input parameters n, r, α, $\Delta d_{min}$ and d are acknowledged by the control in case of the 2nd thread specification method just as in case of the 1st. one. Likewise input parameters can be given with previous block **G76 P** (n) (r) (α) **Q** ($\Delta d_{min}$) **R** (d).
The control refers to block specification made according to the 2nd method if address K is filled out in block.

**Q**: **Angle value** of thread start calculated from encoder zero pulse in °. Interpretation of the address corresponds to that of G33.

**P**: Thread cutting **method**.
Five thread cutting methods can be selected on the basis of the below figures
P1: Cutting amount constant, single edge cutting
P2: Both edge cutting
P3: Cutting depth constant, single edge cutting
P4: Cutting depth constant, both edge cutting
P5: Cutting amount constant, both edge cutting

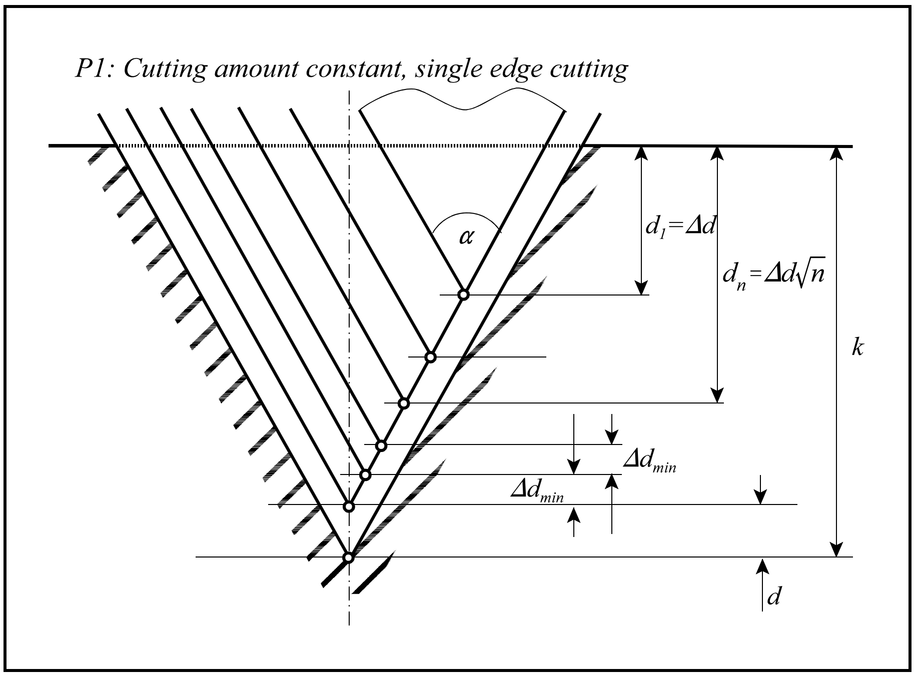


Fig. **17.2.7 -3**

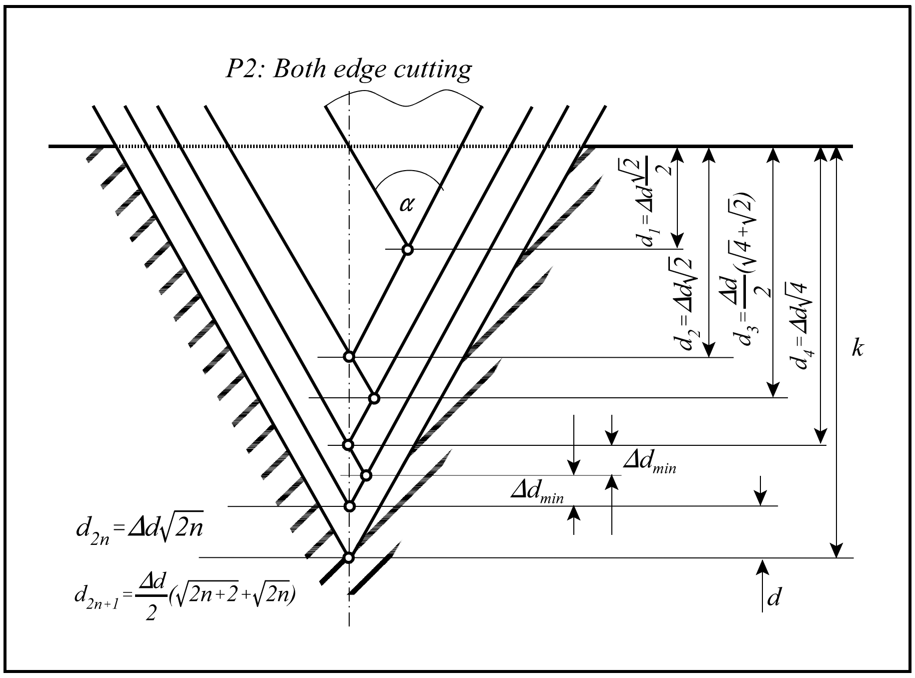

P2: Both edge cutting
$\alpha$
$d_1=\Delta d\frac{\sqrt{2}}{2}$
$d_2=\Delta d\sqrt{2}$
$d_3=\frac{\Delta d}{2}(\sqrt{4}+\sqrt{2})$
$d_4=\Delta d\sqrt{4}$
$k$
$\Delta d_{min}$
$\Delta d_{min}$
$d_{2n}=\Delta d\sqrt{2n}$
$d_{2n+1}=\frac{\Delta d}{2}(\sqrt{2n+2}+\sqrt{2n})$
$d$

Fig. **17.2.7 -4**

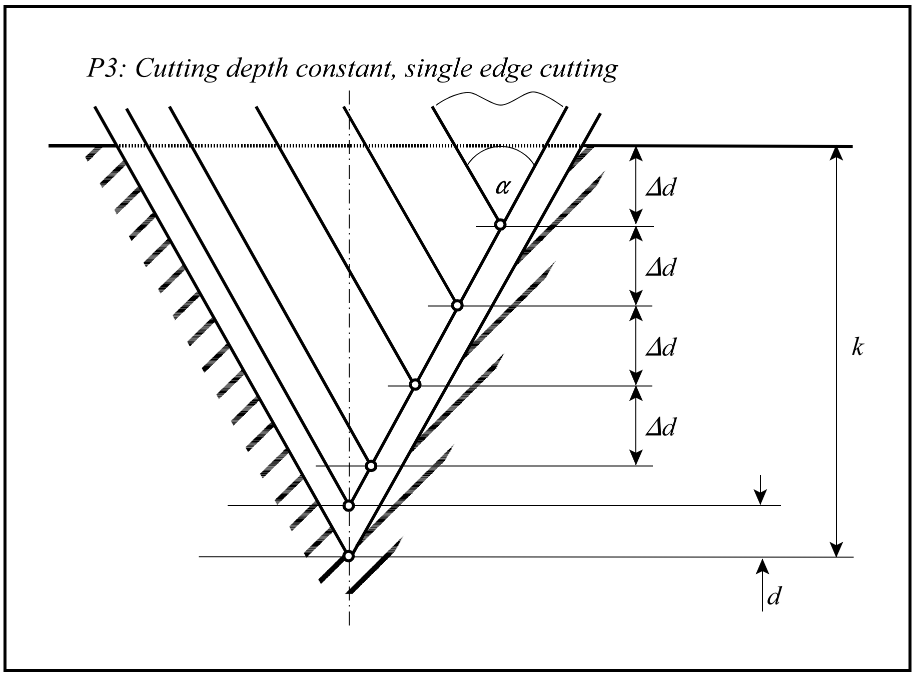


Fig. **17.2.7 -5**

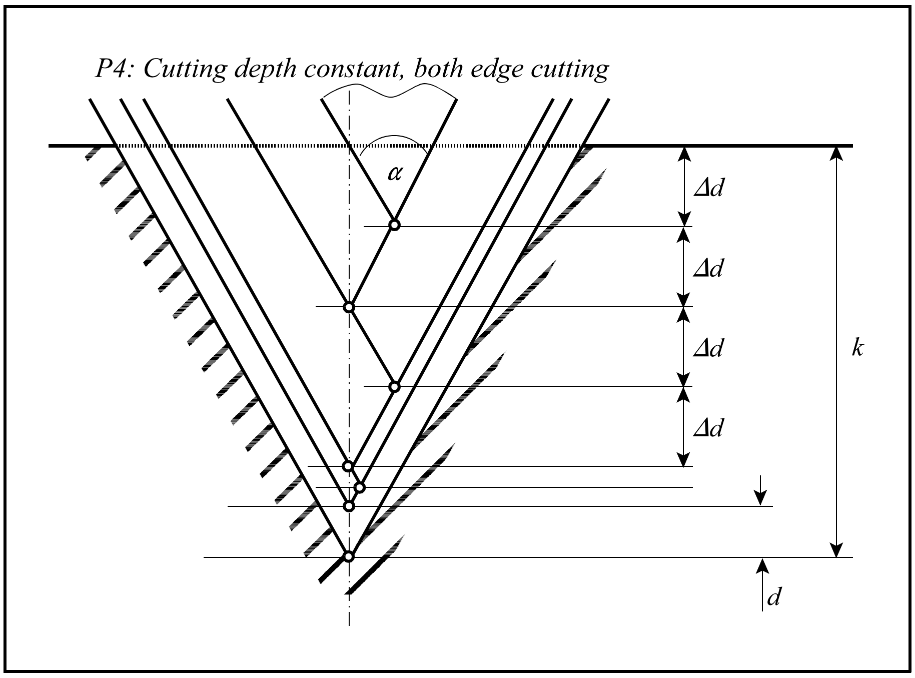


Fig. **17.2.7 -6**

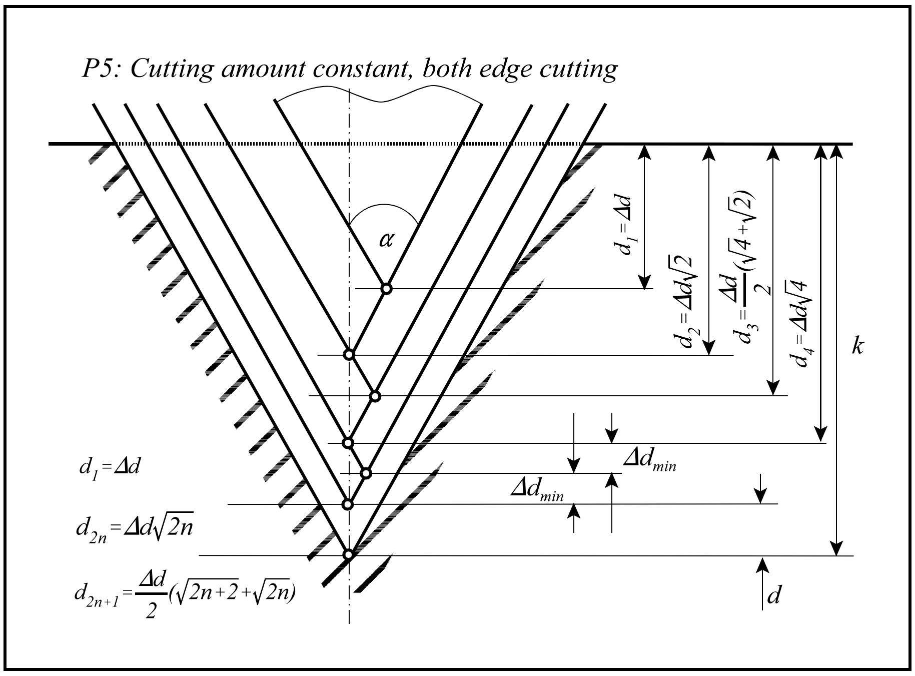

P5: Cutting amount constant, both edge cutting
$\alpha$
$d_1=\Delta d$
$d_2=\Delta d\sqrt{2}$
$d_3=\frac{\Delta d}{2}(\sqrt{4}+\sqrt{2})$
$d_4=\Delta d\sqrt{4}$
k
$\Delta d_{min}$
$\Delta d_{min}$
$d_1=\Delta d$
$d_{2n}=\Delta d\sqrt{2n}$
$d_{2n+1}=\frac{\Delta d}{2}(\sqrt{2n+2}+\sqrt{2n})$
d

Fig. **17.2.7 -7**

# 18 Canned Cycles for Drilling

A drilling cycle may be broken up into the following operations.

Operation 1: Positioning in the Selected Plane
Operation 2: Operation After Positioning
Operation 3: Movement in Rapid Traverse to Point R
Operation 4: Operation in Point R
Operation 5: Drilling
Operation 6: Operation at the Bottom of the Hole
Operation 7: Retraction to Point R
Operation 8: Operation at Point R
Operation 9: Retraction in Rapid Traverse to the Initial Point
Operation 10: Operation at the Initial Point

**Point R, point of approach. -** The tool approaches the workpiece to that point in rapid traverse.
**Initial point.** - The position of the drilling axis assumed prior to the start of cycle.

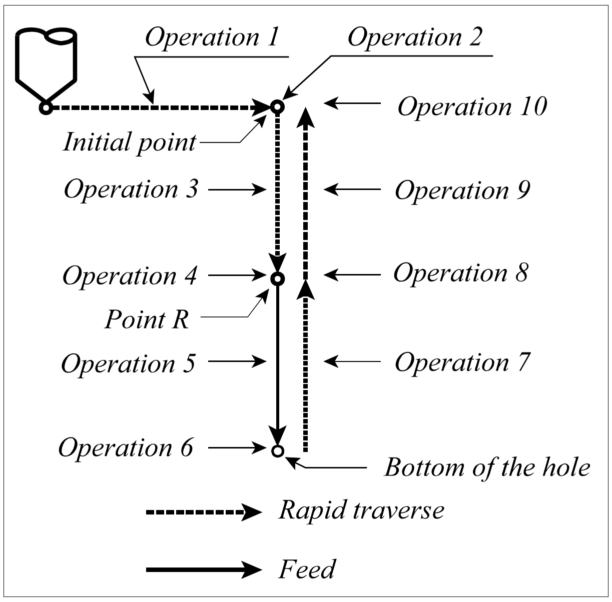


Fig. **18-1**

The above operations give a general picture of a drilling cycle, some of them may be omitted in specific instances.
A drilling cycle has a **positioning plane** and a **drilling axis**. The plane of positioning and the drill axis will be selected by plane selection instructions G17, G18, G19.

| G code | 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 $X_p$ is axis X or the one parallel to it
$Y_p$ is axis Y or the one parallel to it
$Z_p$ is axis Z or the one parallel to it.

Axes U, V, W are regarded to be parallel ones when they are defined in parameters.

***If face drilling is to be programmed, where the drilling axis is Z, plane G17 must be selected, if however side drilling is to be programmed, where the drilling axis is X, plane G19 must be selected.***

The **drilling cycles can be configured** with instructions G98 and G99.

**G98** : The tool is retracted as far as the initial point in the course of the drilling cycle. A normal (default) status assumed by the control after power-on, reset or deletion of cycle mode.

**G99** : The tool is retracted as far as point R in the course of the drilling cycle; accordingly, operations 9 and 10 are omitted.

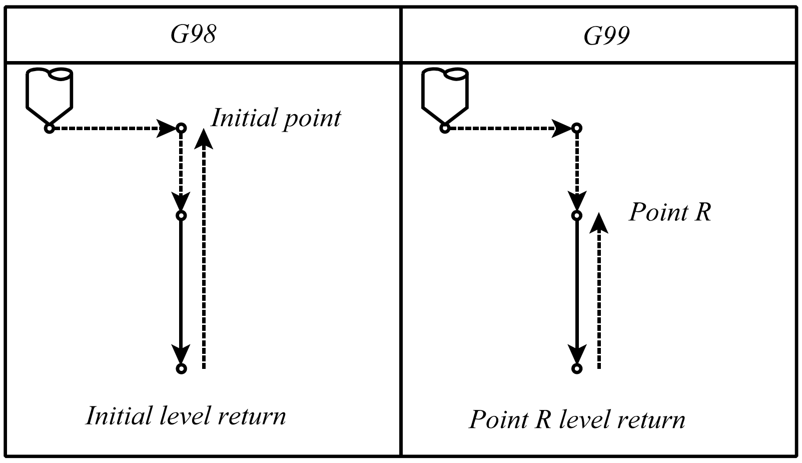


Fig. **18**-2

**Codes of the drilling cycles**: G83.1, G84.1, G86.1, G81, ..., G89
They will set up the particular cycle mode enabling the cycle variables to be modal.
Code G80 willcancel the cycle mode and delete the cycle variables from the memory.

**Addresses** used in the **drilling cycles** (and meanings thereof) are:

```
G17  G_   Xp_ Yp_ C_   I_ J_   Zp_ R_ Q_ E_ P_ F_ S_   L_
G18  G_   Zp_ Xp_ C_   K_ I_   Yp_ R_ Q_ E_ P_ F_ S_   L_
G19  G_   Yp_ Zp_ C_   J_ K_   Xp_ R_ Q_ E_ P_ F_ S_   L_
                                                       └─  No. of repetitions
                                                   └─────  data of drilling
                       └───────── displacement after spindle orientation
              └────────────────── position of hole
     └─────────────────────────── code of drilling
```

The **code of drilling**:

For meanings of the codes see below.
Each code will be modal until an instruction G80 or a code is programmed, that belongs to G code group 1 (interpolation codes: G01, G02, G03, G33).
As long as the cycle state is on (instructions G83.1, G84.1, G86.1, G81,...G89), the modal cycle variables will be modal between drilling cycles of various types, too.

**Initial point**:

The initial point is the position of axis selected for drilling; it will be recorded
– when the cycle mode is set up. For example, in the case of

```
N1 G17 G90 G0 Z200
N2 G81 X0 C0 Z50 R150
N3 X100 C30 Z80
```

the position of initial point will be Z=200 in blocks N2 and N3, too.
– Or when a new drilling axis is selected. For example:

```
N1 G17 G90 G0 Z200 W50
N2 G81 X0 C0 Z50 R150
N3 X100 C30 W20 R25
```

position of start point is Z=200 in block N2
position of start point is W=50 in block N3
Programming of R is mandatory when the selection of drilling axis is changed, or else error message *3053 NO BOTTOM OR R POINT* is returned.

**Position of hole - $X_p$, $Y_p$, $Z_p$, C**

Of the coordinate values entered, those in the selected plane will be taken for the position of the hole.
The values entered may be incremental or absolute ones, rectangular (Cartesian) or polar coordinates in metric or inch units.
The mirror image, coordinate system rotation and scaling commands are applicable to the coordinate values entered.
The control moves to the position of hole in rapid traverse regardless of which code in group1 is in effect.

**Displacement after spindle orientation - I, J, K**

If the particular machine is provided with the facility of spindle orientation, the tool can be retracted off the surface in fine boring and back boring cycles G76 and G87 in order not to scratch the surface of the hole. Now the direction in which the tool is to be withdrawn from the surface can be specified at addresses I, J or K. The control will interpret the addresses in conformity with the plane selected.

| | |
|---|---|
| G17: | I, J |
| G18: | K, I |
| G19: | J, K |

Each address is interpreted as an incremental data of rectangular coordinates. The address may be a metric or inch one.
The mirror image, coordinate system rotation and scaling commands are not applicable to data of I, J, K. The latter are modal values. They are deleted by G80 or by the codes of the interpolation

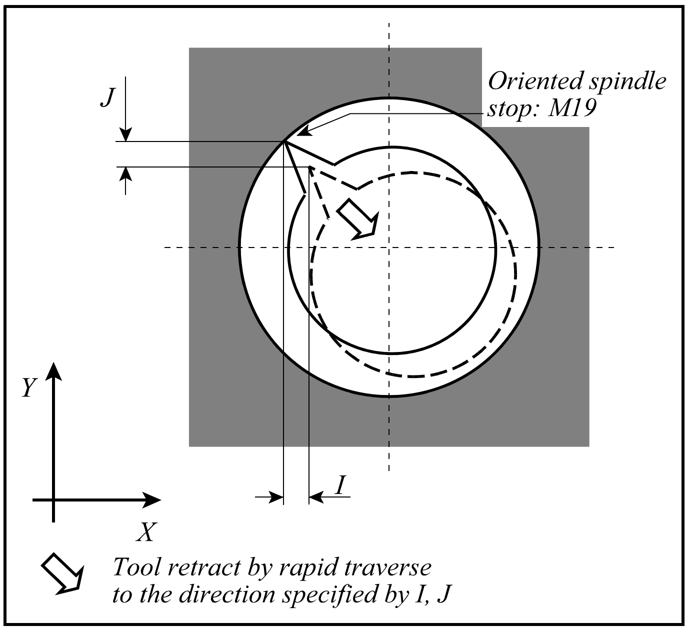

J
Oriented spindle stop: M19
Y
X
I
Tool retract by rapid traverse to the direction specified by I, J

Fig. 18-3

group. Withdrawal is accomplished in rapid traverse.

**Data of drilling**

Bottom position of the hole (point Z): $X_p$, $Y_p$, $Z_p$

The bottom position of the hole or point Z (in case of G17) has to be specified at the address of the drilling axis. The coordinate of the bottom point of the hole will always be interpreted in terms of rectangular data. It may be specified in inch or metric units, absolute or incremental values. When the value of the bottom point is specified incrementally, the displacement will be calculated from point R.

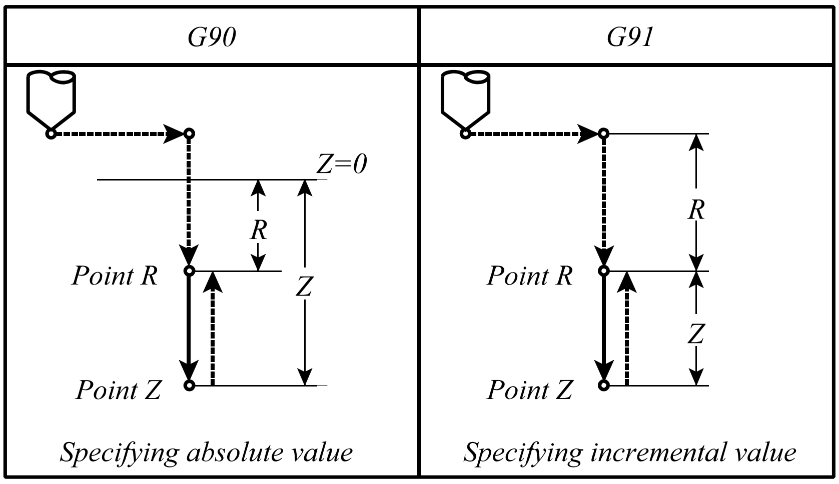


Fig. **18-4**

The mirror image and scaling commands are applicable to the data of the bottom point. The latter are modal values deleted by G80 or by the codes of the interpolation group. The control will always approach point Z with the particular feed in effect.

Point R

The point of approach is specified at address R. It is always a rectangular coordinate data that may be an incremental or absolute one, metric or inches. When data R is an incremental one, its value is calculated from the initial point. The mirror image and scaling data are applicable to the data of point R. They are modal data deleted by G80 or by the codes of the interpolation group. The control will always approach point R in a rapid traverse.

Cut-in value (Q)

It is the depth of the cut-in, in the cycles of G73 and G83. It is invariably an incremental, rectangular positive data (a modal one). Its value will be deleted by G80 or by the codes of the interpolation group. The scaling does not affect the value of cut-in depth.

Auxiliary data (E)

The extent of retraction in the cycle of G73 and value of clearance in the cycle of G83 is specified on address E. It is always an incremental, rectangular, positive data. The scaling command has no effect to the auxiliary data. (Modal value). Its value will be deleted by G80 or by the codes of the interpolation group. Unless it has been programmed, the control will take the necessary value from parameter *RETG73*, or *CLEG83*.

Dwell (P)

Specifies the time of dwell at the bottom of the hole. Its specification is governed by the rules described at G04. The value of the dwell is a modal one deleted by G80 or by the codes of the interpolation group.

Feed (F)

It will define the feed. A modal value, re-written only by the programming of another data F. It will not be deleted by G80 or some other code.

Spindle speed (S)

A modal value re-written only by programming another data S. It will not be deleted by G80 or some other code.

Number of repetitions (L)

Defining the number of cycle repetitions over the range of 1 through 9999. Unless L is filled in, the value of L=1 is assumed. In the case of L=0 the data of the cycle will be stored but not executed. The value of L is effective only in the block in which it has been specified.

**Examples of modal drilling codes and cycle variables**:

```
N1 G17 G0 Z_ M3
N2 G81 X_ C_ Z_ R_ F_
```

It is mandatory to specify the data of drilling (Z, R) at the beginning of cycle mode.

```
N3 X_
```

Since the data of drilling have been specified in block N2 and they are used unchanged in N3, they need not be filled in again, i.e., G81, Z_, R_, F_ may be omitted. The position of the hole is varied in direction X only, the tool moves in that direction and will drill the same hole as in block N2.

```
N4 G82 C_ Z_ P_
```

The position of the hole is shifted in direction C. The method of drilling complies with G82, the bottom point assumes a new value (Z), the point R and the feed (R, F) are taken from block N2.

```
N5 G80 M5
```

The cycle mode and the modal cycle variables (except for F) will be deleted.

```
N6 G85 C_ Z_ R_ P_ M3
```

Since the data of drilling are deleted in block N5 under the command of G80, the values of Z, R and P have to be specified again.

```
N7 G0 X_ C_
```

The cycle mode and the modal cycle variables (except for F) will be deleted.

**Examples of using cycle repetitions**:

If a particular type of hole is to be drilled with unchanged parameters at equally spaced positions, the number of repetitions can be specified at address L. The value of L is only effective in the block, in which it has been specified.

```
N1 G90 G19 G0 X300 Z40 C0 M3
N2 G91 G81 X-40 Z100 R-20 F50 L5
```

Under the above instructions the control will drill 5 identical holes spaced at 100mm along axis Z. The position of the first hole is Z=140, C=0. Since this is a side drilling with X axis, plane G19 has been selected.

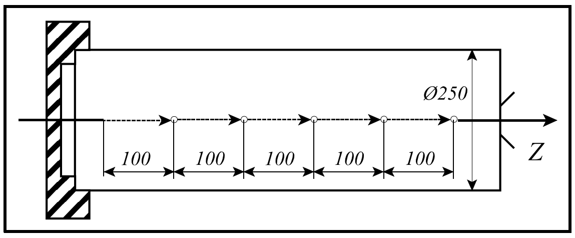


Fig. 18-5

Under G91 the position of the hole has been specified incrementally.

```
      N1 G90 G17 G0 X200 C–60 Z50
      N2 G81 CI60 Z–40 R3 F50 L6
```

Under the above commands the control will drill 6 holes spaced at 60 degrees around a 100mm-radius circle. The position of the first hole coincides with the point of X=200 C=0 coordinates. Since it is face drilling (the drill axis is Z) plane G17 is specified.

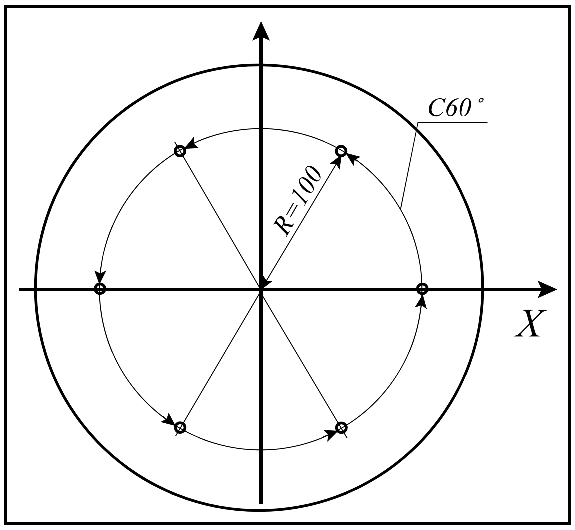


Fig. **18-6**

## 18.1 Detailed Description of Canned Cycles

### 18.1.1 High Speed Peck Drilling Cycle (G83.1)

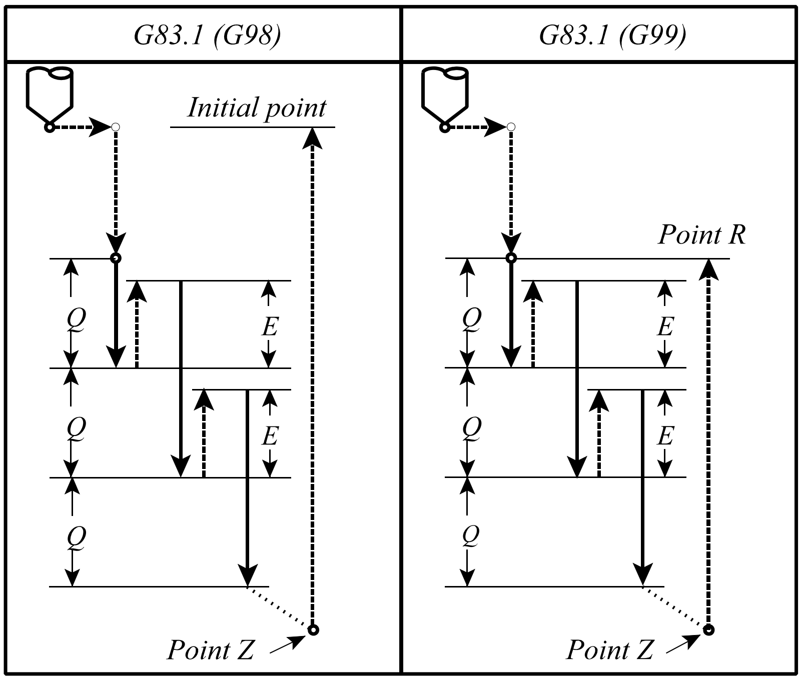


Fig. **18.1.1**-1

The variables used in the cycle are

G17 **G83.1** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ Q__ E__ F__ L__
G18 **G83.1** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ Q__ E__ F__ L__
G19 **G83.1** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ Q__ E__ F__ L__

The operations of the cycle are

1. rapid-traverse positioning
2. -
3. rapid-traverse movement to point R
4. -
5. drilling as far as the point Z, with feed F
6. -
7. with G99, retraction to point R, in rapid traverse
8. -
9. with G98, retraction to the initial point, in rapid traverse
10. -

Description of drilling operation 5 is as follows:
– Drilling the cut-in depth specified at address Q in the workpiece, with feed,
– rapid-traverse retraction by the distance specified at address E or in parameter *RETG73,*
– drilling cut-in depth Q again, reckoned from the end point of the previous cut-in,
– rapid-traverse retraction at the value specified at address E.
The above procedure is carried on as far as the bottom point specified at address Z.

### 18.1.2 Counter Tapping Cycle (G84.1)

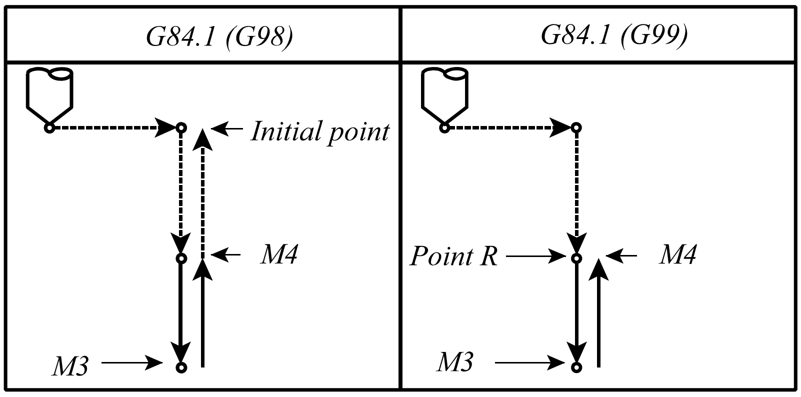


Fig. **18.1.2-1**

This cycle can be used only with a spring tap. The variables used in the cycle are

G17 **G84.1** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ (P__) F__ L__
G18 **G84.1** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ (P__) F__ L__
G19 **G84.1** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ (P__) F__ L__

Prior to start the cycle, the spindle has to be started or programmed to rotate in the direction of M4 (counter-clockwise).
The value of feed has to be specified in conformity with the thread pitch of the tapper.
– In state G94 (feed per minute):

$$F=P\cdot S$$

where P is the thread pitch in mm/rev or inches/rev,
S is the spindle speed in rpm
– In state G95 (feed per revolution):

$$F=P$$

where P is the thread pitch in mm/rev or inches/rev.
The operations of the cycle:

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement as far as point R
4. -
5. drilling as far as thebottom point, with feed F (override and stop inhibited)
6. dwell with the value specified at address P, provided parameter *TAPDWELL* is enabled (=1) spindle direction reversal (M3)
7. retraction as far as point R with feed F (override and stop inhibited)
8. spindle direction reversal (M4)
9. with G98, rapid-traverse retraction to the initial point
10. -

### 18.1.3 Fine Boring Cycle (G86.1)

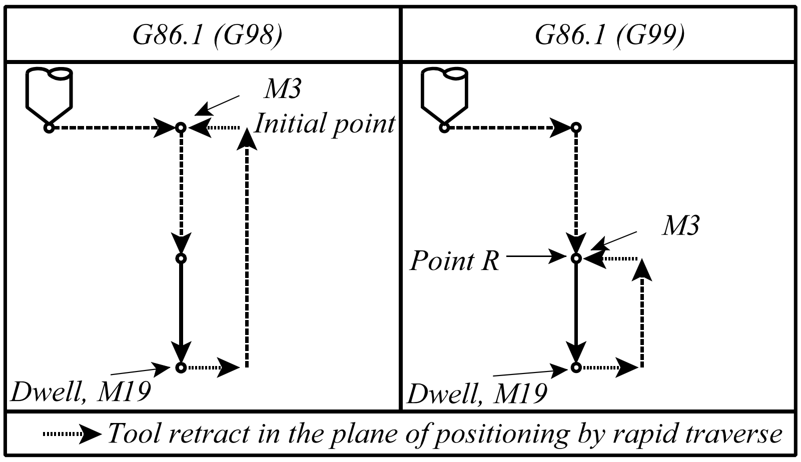


Fig. **18.1.3-1**

Cycle G76 is only applicable when the facility of spindle orientation is incorporated in the machine-tool. In this case parameter ORIENT1 is to be set to 1, otherwise message *3052 ERROR IN G76* is returned.

Since, on the bottom point, the cycle performs spindle orientation and recesses the tool from the surface with the values specified at I, J and K, the part will not be scratched when the tool is withdrawn.

The variables used in the cycle are

G17 **G86.1** $X_p$__ $Y_p$__ C__ I__ J__ $Z_p$__ R__ P__ F__ L__
G18 **G86.1** $Z_p$__ $X_p$__ C__ K__ I__ $Y_p$__ R__ P__ F__ L__
G19 **G86.1** $Y_p$__ $Z_p$__ C__ J__ K__ $X_p$__ R__ P__ F__ L__

Command M3 has to be issued prior to starting the cycle.

The operations of the cycle:

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement as far as point R
4. -
5. boring as far as the point Z, with feed F
6. – dwell with the value specified at address P
   – spindle orientation (M19)
   – rapid-traverse receding of the tool with values I, J, K in the selected plane
7. with G99, rapid-traverse retraction as far as point R
8. with G99,
   – rapid-traverse retraction of the tool in the selected plane, opposite to the values specified at I, J, K
   – spindle re-started in direction M3
9. with G98, rapid-traverse retraction to the initial point
10. with G98,
    – rapid-traverse retraction of the tool in the selected plane, opposite to the values specified at I, J, K
    – spindle re-started in direction M3

### 18.1.4 Canned Cycle for Drilling Cancel (G80)

The code **G80** will cancel the cycle state, the cycle variables will be deleted.
Z and R will assume incremental 0 value (the rest of variables will assume 0).
With coordinates programmed in block G80 but no other instruction is issued, the movement will be carried out according to the interpolation code in effect prior to activation of the cycle (G code group 1).

### 18.1.5 Drilling, Spot Boring Cycle (G81)

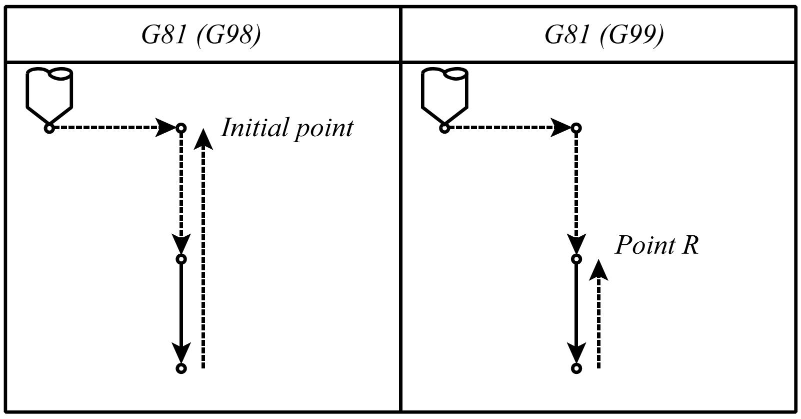


Fig. **18.1.5-1**

The variables used in the cycle are

G17 **G81** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ F__ L__
G18 **G81** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ F__ L__
G19 **G81** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ F__ L__

The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement as far as point R
4. -
5. drilling as far as the point Z, with feed F
6. -
7. with G99, retraction to point R, in rapid traverse
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

### 18.1.6 Drilling, Counter Boring Cycle (G82)

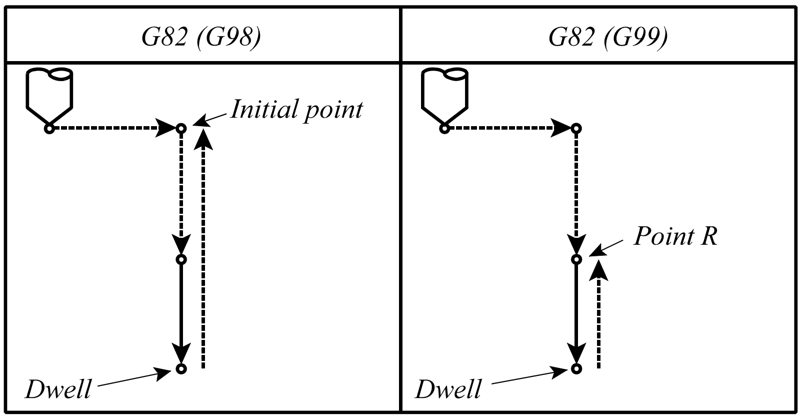


Fig. **18.1.6-1**

The variables used in the cycle are

G17 **G82** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ P__ F__ L__
G18 **G82** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ P__ F__ L__
G19 **G82** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ P__ F__ L__

the operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement as far as point R
4. -
5. drilling as far as the bottom point, with feed F
6. dwell for the time specified at address P
7. with G99, rapid-traverse retraction to point R
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

### 18.1.7 Peck Drilling Cycle (G83)

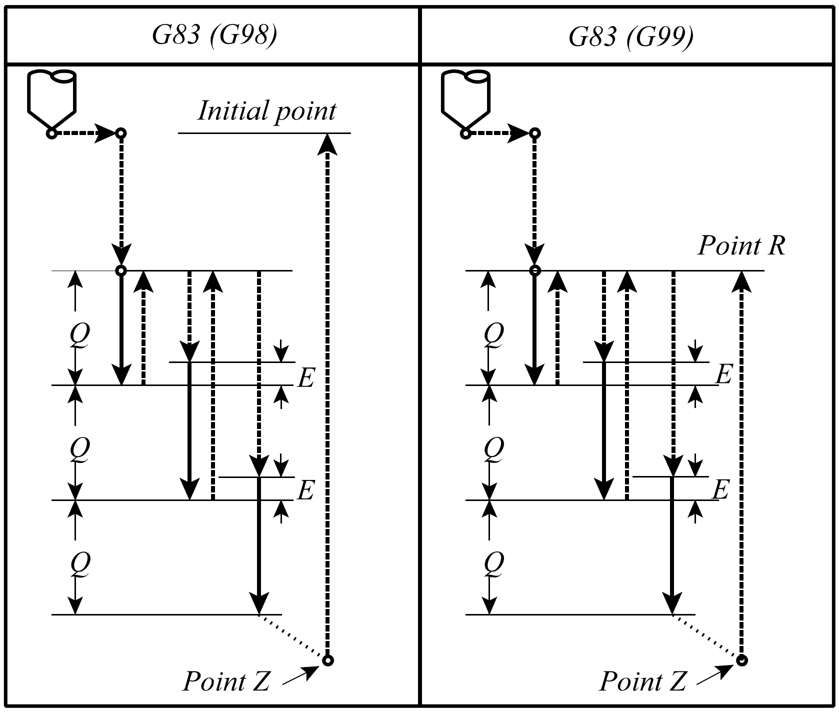


Fig. **18.1.7-1**

The variables used in the cycle are

G17 **G83** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ Q__ E__ F__ L__
G18 **G83** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ Q__ E__ F__ L__
G19 **G83** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ Q__ E__ F__ L__

The oprations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. drilling to the bottom point, with feed F
6. -
7. with G99, rapid-traverse retraction to point R
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

Description of drilling operation 5 is as follows:
– Drilling the depth specified at address Q, with feed,
– rapid-traverse retraction to point R,
– rapid-traverse approach of the previous depth as far as the clearance amount specified on address E,
– drilling depth Q again, reckoned from the previous cut-in, with feed F (displacement E+Q),
– rapid-traverse retraction to point R.

The above procedure is carried on as far as the bottom point specified at address Z. Distance E will be taken from the program (address E) or from parameter *CLEG83*.

### 18.1.8 Tapping Cycle (G84)

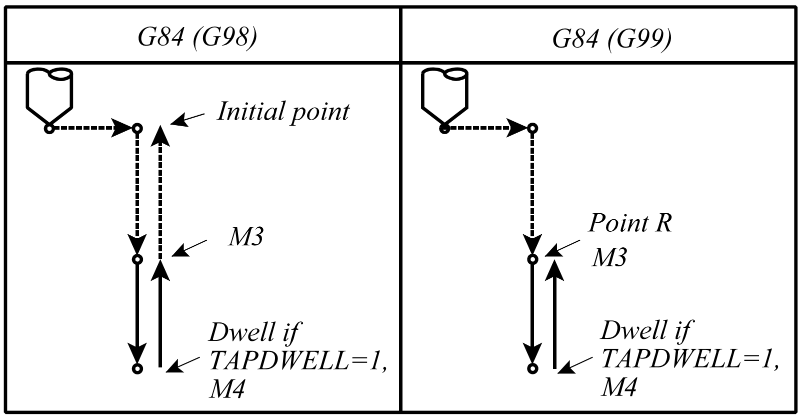


Fig. **18.1.8-1**

This cycle can be used only with a spring tap.
The variables used in the cycle are

G17 **G84** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ (P__) F__ L__
G18 **G84** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ (P__) F__ L__
G19 **G84** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ (P__) F__ L__

Direction of spindle rotation M3 (clockwise) has to be selected prior to starting the cycle.
The value of feed has to be specified in conformity with the thread pitch of the tap.
– In state G94 (feed per minute):

$$F = P \cdot S$$

where P is the thread pitch in mm/rev or inches/rev.
S is the spindle speed in rpm

– In state G95 (feed per revolution):

$$F = P$$

where P is the thread pitch in mm/rev or inches/rev.

The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. tapping to the bottom point with feed F, override and stop inhibited
6. – dwell with value specified at address P, provided parameter *TAPDWELL* is enabled (=1),
   – reversal of spindle direction (M4)
7. retraction to point R with feed F, override and stop inhibited
8. reversal of spindle direction (M3)
9. with G98, rapid-traverse retraction to the initial point
10. -

### 18.1.9 Rigid (Clockwise and Counter-clockwise) Tap Cycles (G84.2, G84.3)

In a tapping cycle the quotient of the drill-axis feed and the spindle rpm must be equal to the thread pitch of the tap. In other words, under ideal conditions of tapping, the quotient

$P=\frac{F}{S}$ must be constant from moment to moment

where P is the thread pitch (mm/rev or inches/rev),
F is the feed (mm/minute or inches/minute),
S is the rpm of spindle (revolutions/minute).

The spindle speed and the feed of the tapping axis are controlled completely independently in left-hand and right-hand tapping cycles G74 and G84, respectively. Accordingly, the above condition cannot be fulfilled to full accuracy. This is particularly applicable to the bottom of the hole where the feed of the drill axis and the spindle speed ought to be slowed down and stopped in synchronism, and accelerated so in the opposite direction. This condition cannot be fulfilled from a controlled point of view in the above case. The above problem can be eliminated by a spring tap, that would compensate for the fluctuations in the value of quotient $\frac{F}{S}$.

A different principle of control is adopted in drilling cycles G84.2 and G84.3 enabling rigid tap (tapping without spring). There the control maintains quotient $\frac{F}{S}$ constant from moment to moment.

The control will regulate only the speed of the spindle in the former case, in the latter case its position is also controlled. The movements of the drilling axis and the spindle are linked through linear interpolations in cycles G84.2 and G84.3. In this way quotient $\frac{F}{S}$ can be maintained constant in the acceleration and deceleration stages as well.

**G84.2**: Rigid tapping cycle
**G84.3**: Rigid counter tapping cycle

The above cycles are only applicable with machines in which the spindle is fitted with an encoder, and the main drive can be fed back for position control (parameter *INDEX1*=1). Otherwise the control will return error message *3052 ERROR IN G76, G87* when the mode is called.

The variables used in the cycle are

G17 **G84._** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ F__ S__ L__
G18 **G84._** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ F__ S__ L__
G19 **G84._** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ F__ S__ L__

The spindle comes to a halt at the end of the cycle, if necessary, it has to be re-started by the programmer.

The feed and the spindle rpm have to be specified in conformity with the thread pitch of the tap.

– In state G94 (feed per minute), $F=P\cdot S$

where P is the thread pitch in mm/rev or inches/rev,
S is the spindle speed in rpm

In this case the displacement and the feed along the drilling axis and the spindle will be as follows (Z assumed to be the drilling axis):

| displacement | feed |
|---|---|
| Z *z= distance between point R and point Z* | $F_z=F\left(\frac{mm}{\min} \text{ or } \frac{inch}{\min}\right)$ |
| S $s=\frac{z\cdot S\cdot 360}{F}$ $(degrees)$ | $F_s=S\cdot 360\left(\frac{degrees}{\min}\right)$ |

– In state G95 (feed per revolution), $F=P$

where P is the thread pitch in mm/rev or inches/rev. Evidently, the thread pitch can be programmed directly in state G95 (feed per revolution), but S also has to be programmed in order to define the feed.

In this case, the displacement and the feed along the drilling axis and the spindle will be as follows (assuming axis Z to be the drilling axis):

| displacement | feed |
|---|---|
| Z *z=R distance between point and the base point* | $F_z=F\cdot S\left(\frac{mm}{\min} \text{ or } \frac{inch}{\min}\right)$ |
| S $s=\frac{z\cdot 360}{F}$ $(degrees)$ | $F_s=S\cdot 360\left(\frac{degrees}{\min}\right)$ |

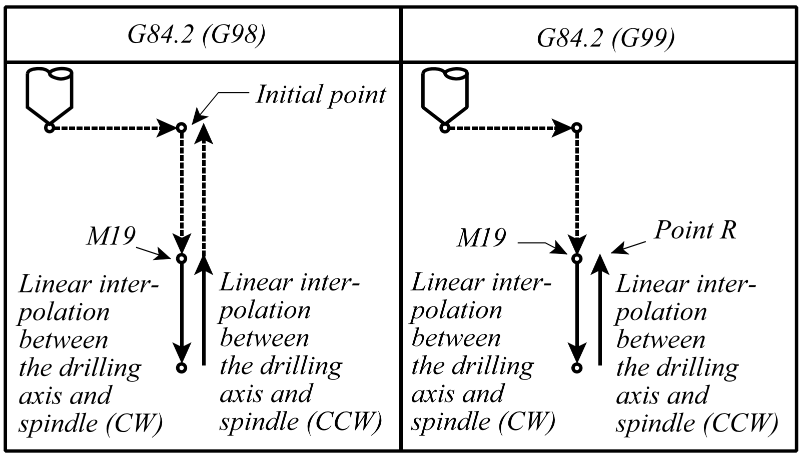

G84.2 (G98)
G84.2 (G99)
Initial point
M19
Linear interpolation between the drilling axis and spindle (CW)
Linear interpolation between the drilling axis and spindle (CCW)
M19
Point R
Linear interpolation between the drilling axis and spindle (CW)
Linear interpolation between the drilling axis and spindle (CCW)

Fig. **18.1.9**-1

In the case of G84.2, the operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. spindle orientation (M19)

5. linear interpolation between the drilling axis and the spindle, with the spindle rotated in clockwise direction
6. -
7. linear interpolation between the drilling axis and the spindle, with the spindle being rotated counter-clockwise
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

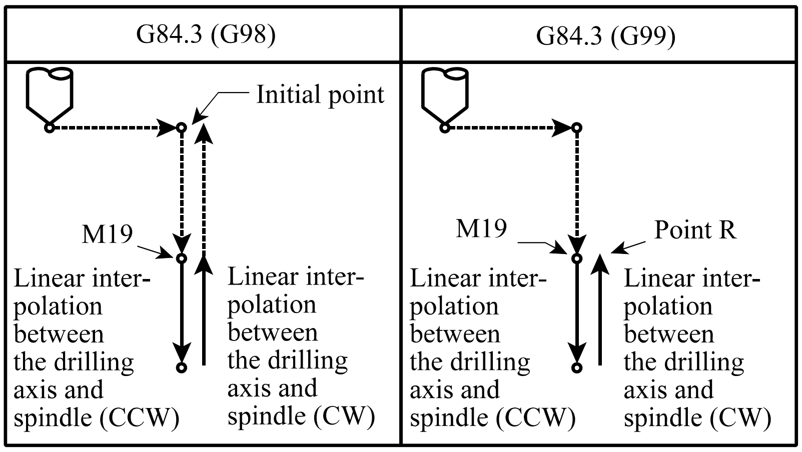

G84.3 (G98)
G84.3 (G99)
Initial point
M19
Linear interpolation between the drilling axis and spindle (CCW)
Linear interpolation between the drilling axis and spindle (CW)
M19
Point R
Linear interpolation between the drilling axis and spindle (CCW)
Linear interpolation between the drilling axis and spindle (CW)

Fig. **18.1.9-2**

In the case of G84.3, the operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. spindle orientation (M19)
5. linear interpolation between the drilling axis and the spindle, with the spindle rotated in counter-clockwise direction (-)
6.
7. linear interpolation between the drilling axis and the spindle, with the spindle being rotated clockwise (+)
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

### 18.1.10 Boring Cycle (G85)

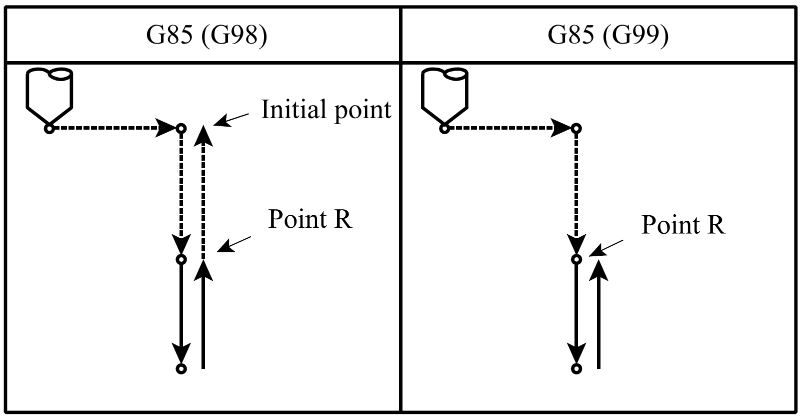


Fig. **18.1.10-1**

The variables used in the cycle are

G17 **G85** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ F__ L__
G18 **G85** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ F__ L__
G19 **G85** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ F__ L__

The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. boring as far as the bottom point with feed F
6. -
7. retraction to point R with feed F
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

## 18.1.11 Boring Cycle Tool Retraction with Rapid Traverse (G86)

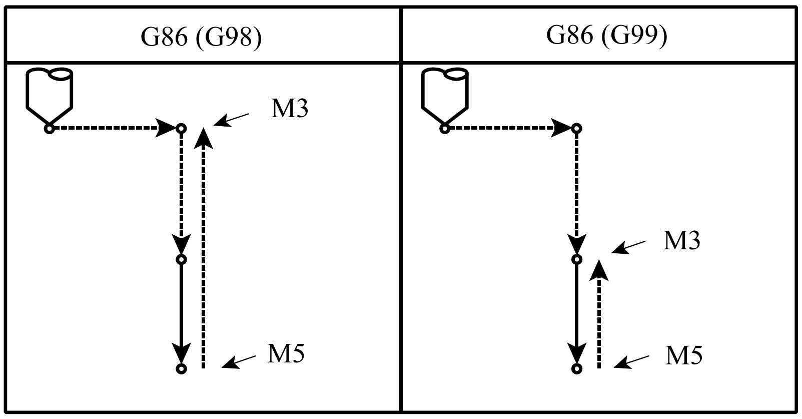


Fig. **18.1.11-1**

The variables used in the cycle are

G17 **G86** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ F__ L__
G18 **G86** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ F__ L__
G19 **G86** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ F__ L__

The spindle has to be given rotation of M3 when the cycle is started.
The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. boring as far as the point Z with feed F
6. stopping the spindle (M5)
7. with G99, rapid-traverse retraction to point R
8. with G99, spindle re-started (M3)
9. with G98, rapid-traverse retraction to the start point
10. with G98, spindle re-started (M3)

### 18.1.12 Boring Cycle/Back Boring Cycle (G87)

The cycle will be performed in two different ways.

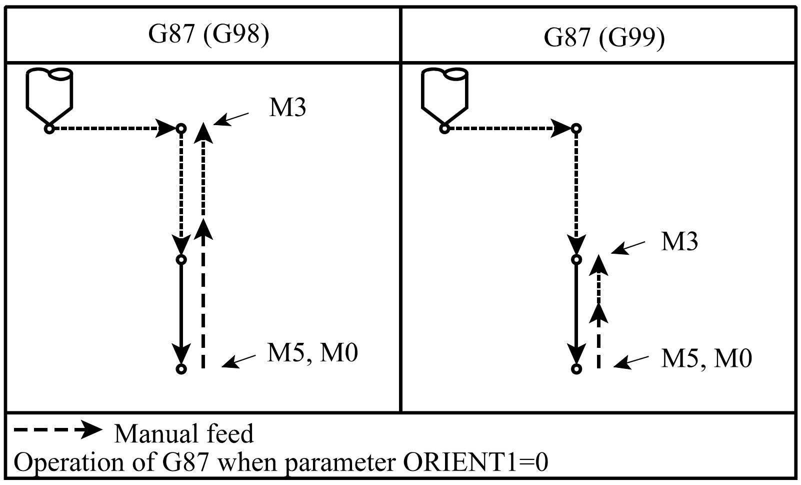


Fig. **18.1.12-1**

**A. Boring Cycle, Manual Operation at Bottom Point**

Unless the machine is provided with the facility of spindle orientation (parameter *ORIENT1*=0), the control will act according alternative "A".

The variables used in the cycle are

G17 **G87** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ F__ L__
G18 **G87** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ F__ L__
G19 **G87** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ F__ L__

The spindle must be started in M3 when the cycle is started.

The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. boring as far as the bottom point with feed F
6. – spindle stop (M5)
   – the control assumes STOP state (M0), from which the operator can get in one of the manual movement modes (JOG, INCREMENTAL JOG, or HANDLE) and operate the machine manually, for example retract the tool from the side of the hole then remove the tool from the hole. After returning AUTO mode machining can be continued by START.
7. with G99, START followed by rapid-traverse retraction to point R
8. with G99, spindle re-started (M3)
9. with G98, START followed by rapid-traverse retraction to the initial point
10. with G98, spindle re-started (M3)

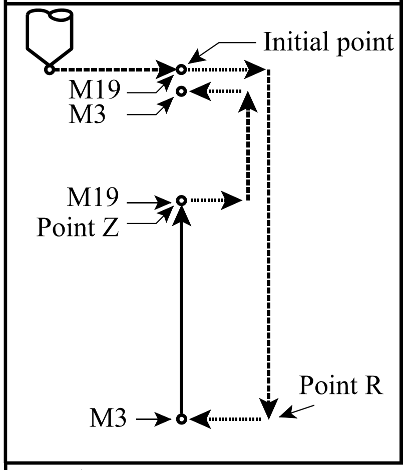
| G87 (G98) | G87 (G99) |
|---|---|
|  Initial point<br>M19<br>M3<br>M19<br>Point Z<br>Point R<br>M3 | Not used |
| ┈➤ Tool retract in the plane of positioning by rapid traverse<br>Operation of G87 when parameter ORIENT1=1 | |

Fig. **18.1.12-2**

**B. Back Boring Cycle**

If the machine is provided with the facility of spindle orientation (parameter *ORIENT1*=1), the control will act in conformity with case "B".

The variables of cycle are

G17 **G87** $X_p$__ $Y_p$__ C__ I__ J__ $Z_p$__ R__ F__ L__
G18 **G87** $Z_p$__ $X_p$__ C__ K__ I__ $Y_p$__ R__ F__ L__
G19 **G87** $Y_p$__ $Z_p$__ C__ J__ K__ $X_p$__ R__ F__ L__

The spindle must be given rotation M3 when the cycle is started.

The operations of cycle are

1. rapid-traverse positioning in the selected plane
2. – spindle orientation
   – tool receded in the selected plane with values I, J, K (rapid traverse)
3. rapid-traverse movement to point R
4. – tool receded in the selected plane opposite to the values specified at I, J or K (rapid traverse)
   – spindle re-started in direction M3
5. boring as far as the point Z, with feed F
6. – spindle orientation (M19)
   – tool receded in the selected plane with values I, J, K (rapid traverse)
7. -
8. -
9. rapid-traverse retraction to the initial point
10. – tool receded in the selected plane opposite to the values specified at I, J or K (rapid traverse)
    – spindle re-started in direction M3

Following from the nature of the cycle, point R is located, unlike in the previous instances, lower than point Z. This must be taken into account in programming the boring axis and addresses R.

## 18.1.13 Boring Cycle (Manual Operation on the Bottom Point) (G88)

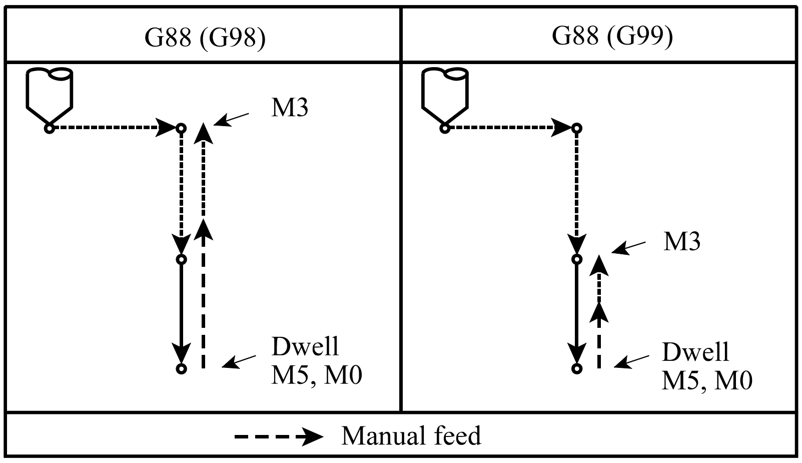


Fig. **18.1.13-1**

The variables used in the cycle are

G17 **G88** $X_p$__ $Y_p$__ C__ $Z_p$__ R__ P__ F__ L__
G18 **G88** $Z_p$__ $X_p$__ C__ $Y_p$__ R__ P__ F__ L__
G19 **G88** $Y_p$__ $Z_p$__ C__ $X_p$__ R__ P__ F__ L__

The spindle must be given rotation M3 when the cycle is started.
The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. boring as far as the bottom point with feed F
6. – dwell with value P
   – spindle stop (M5)
   – the control assumes STOP state (M0), from which the operator can get in one of the manual movement modes (JOG, INCREMENTAL JOG, or HANDLE) and operate the machine manually, for example retract the tool from the side of the hole then remove the tool from the hole. After returning AUTO mode machining can be continued by START.
7. with G99, START followed by retraction to point R (rapid traverse)
8. with G99, spindle re-started (M3)
9. with G98, rapid-traverse retraction to the initial point
10. with G98, spindle re-started (M3)

The cycle is the same as case "A" of G87 but dwelling before the spindle stop.

### 18.1.14 Boring Cycle (Dwell on the Bottom Point, Retraction with Feed) (G89)

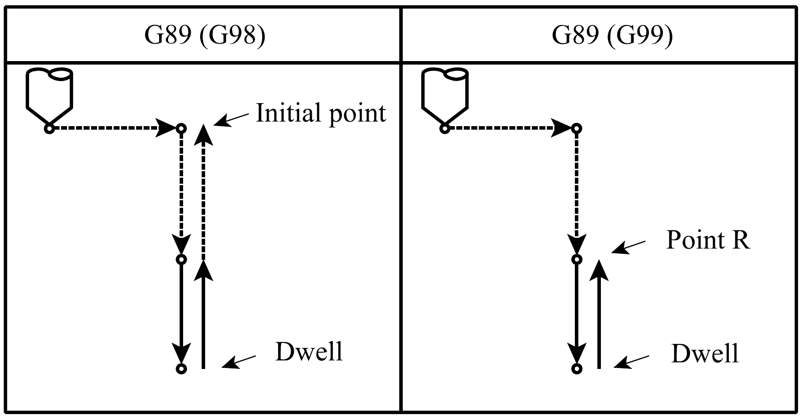


Fig. **18.1.14-1**

The variables used in the cycle are

G17 **G89** $X_p$__ $Y_p$__ $Z_p$__ R__ P__ F__ L__
G18 **G89** $Z_p$__ $X_p$__ $Y_p$__ R__ P__ F__ L__
G19 **G89** $Y_p$__ $Z_p$__ $X_p$__ R__ P__ F__ L__

The operations of the cycle are

1. rapid-traverse positioning in the selected plane
2. -
3. rapid-traverse movement to point R
4. -
5. boring as far as the bottom point, with feed F
6. dwelling with the value specified at address P
7. retraction to point R, with feed F
8. -
9. with G98, rapid-traverse retraction to the initial point
10. -

Except for dwelling, the cycle is identical with G85.

## 18.2 Notes to the Use of Canned Cycles for Drilling

– The drilling cycle will be executed in cycle mode provided a block without code G contains one of the addresses

$X_p$, $Y_p$, $Z_p$, C or R

Otherwise, the drilling cycle will not be executed.

– With dwell G04 P programmed in cycle mode, the command will be executed in conformity with P programmed, but the cycle variable of dwell will **not** be deleted and will **not** be re-written.

– The values of I, J, K, Q, E, P have to be specified in a block, in which drilling is also performed, or else the values will **not** be stored.

To illustrate the foregoing, let us see the following example.

```
G81 X_ C_ Z_ R_ F     (the drilling cycle is executed)
    X                 (the drilling cycle is executed)
    F_                (the drilling cycle is not executed, F is over-
                      written)
    M_                (the drilling cycle is not executed, code M is
                      executed)
 G4 P_                (the drilling cycle is not executed, the dwell
                      will be re-written, but not the dwell value of
                      cycle variable)
    I_ Q_             (the  drilling  cycle  is  not  executed, the
                      programmed values will not be recorded as cycle
                      variables)
```

– If a function as well as a drilling cycle are programmed in one block, the function will be executed at the end of the first operation, on completion of positioning. If L has also been programmed in the cycle, the function will be executed in the first round only.

– In block-by- block mode, the control will stop after each of operations 1, 3 and 10 during the cycle.

– The STOP button is ineffective to each of operations 5, 6 and 7 of cycles G84.1, G84. If STOP is depressed during those operations, the control will continue its functioning, and will not stop before the end of operation 7.

– The feed and the spindle override will always be 100% in each of operations 5, 6 and 7 of cycles G74, G84 regardless of the override switch setting.

– If Tnnmm is programmed in cycle block the new length compensation is taken into account in case of both for positioning in the selected plane and for drilling.

# 19 Polygonal Turning

In case of polygonal turning both the tool and the workpiece are rotated in relation to each other by a specified revolution ratio. Polygons with varying side number are resulted from the change of the revolution ratio and the number of cutters of the rotating tool. The side number of the polygon is determined by the multiplication of the revolution ratio and of the number of cutters. If for example the revolution ratio of tool and work is 2:1 and the number of cutters is 2 a square is turned, while if the latter is 3 a hexagon is turned. This way, e.g., hexagon bolts or nuts can simply be machined by the use of turning. This machining is by far faster than the milling of a polygon by means of polar coordinate interpolation, however the resulting sides are not exact planes.

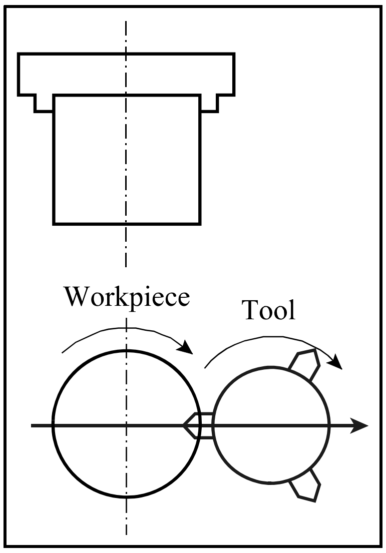


Fig. **19-1**

## 19.1 Principle of Polygonal Turning

The principle of polygonal turning is discussed below.
Suppose, that the rotary axis of the work is on coordinate X=0, Y=0.
Suppose that the rotary axis of the tool is point $P_0$ the coordinates of which are: X=A, Y=0, i.e., the distance between the rotary axis of the tool and that of the workpiece is A.
Suppose, that the radius of the rotating tool is B. The tool tip is indicated by $P_t$. The coordinate of the tool tip is at t=0 moment X=A–B; Y=0.
Assume, that the angular speed of workpiece rotation is α, while that of tool rotation is β.

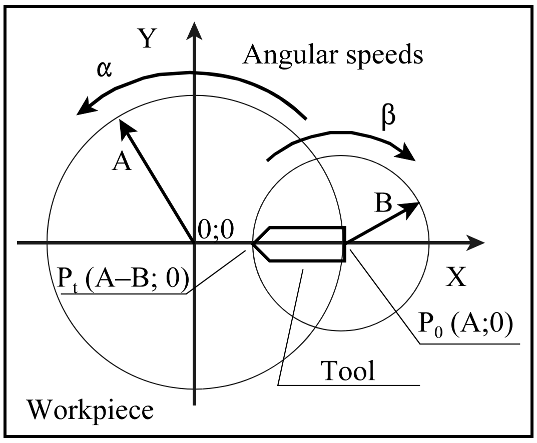


Fig. **19.1-1**

Coordinates of tool tip at t time $P_t$ ($x_t$ ; $y_t$) are the followings:

$$x_t = A\cos\alpha t - B\cos(\beta - \alpha)t$$
$$y_t = A\sin\alpha t + B\sin(\beta - \alpha)t$$

If the revolution ratio of the tool and the workpiece is assumed to be 2:1, then β=2α. Put it in the place of the above equation:

$$x_t = A\cos\alpha t - B\cos\alpha t = (A - B)\cos\alpha t$$
$$y_t = A\sin\alpha t + B\sin\alpha t = (A + B)\sin\alpha t$$

The above equation results in an ellipse, the length of the major axis of which is A+B, while that of the minor axis is A–B.
In case the tools lie symmetrically 180° from each other a square is turned, in case they are placed 120° from each other a hexagon is turned, on condition that the revolution ratio of tool and workpiece is 2:1.

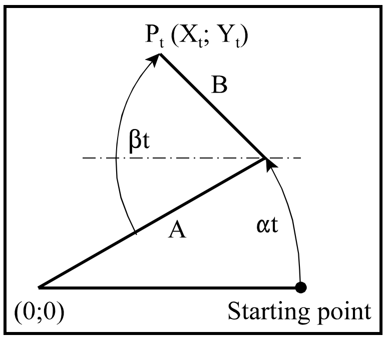


Fig. **19.1-2**

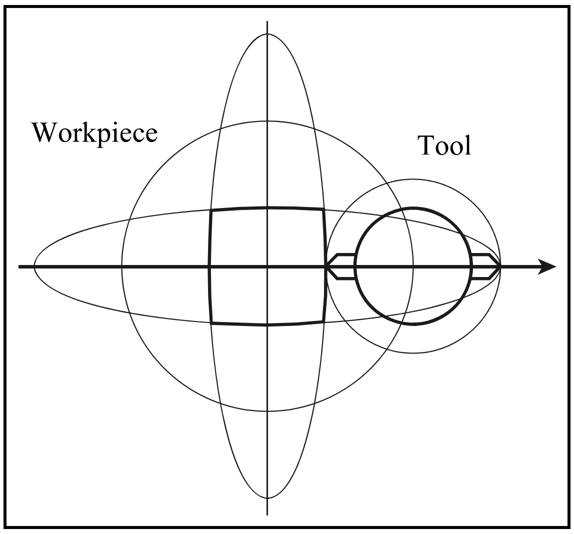


Fig. **19.1-3**

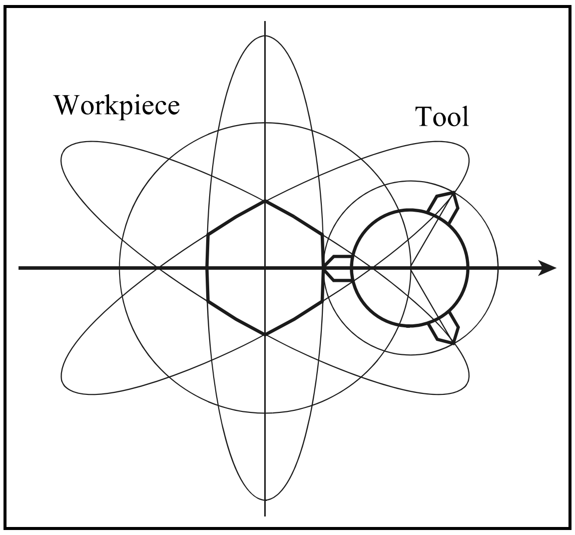


Fig. **19.1-4**

With other revolution ratios the curves will differ from ellipses, although the polygonal sides can be estimated even with those forms.

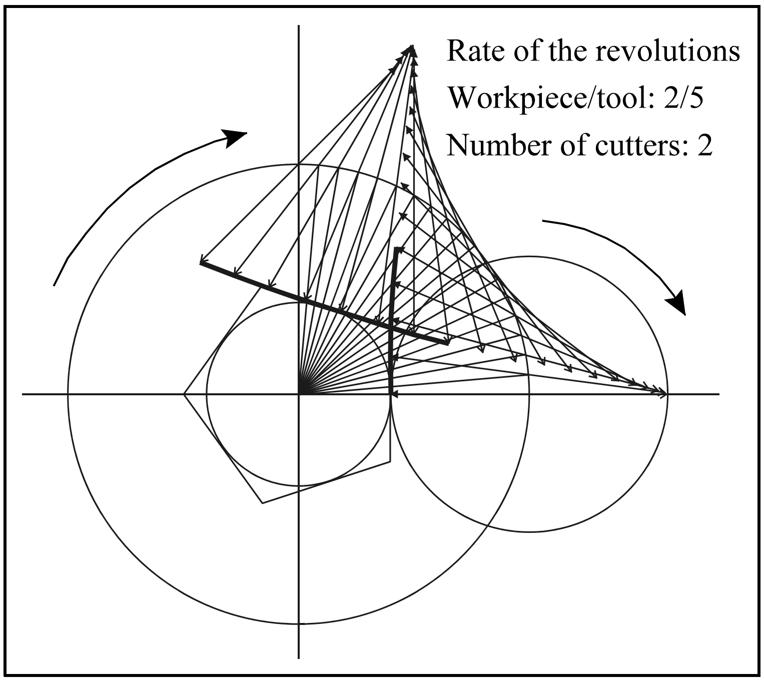


Fig. **19.1-5**

## 19.2 Programming Polygonal Turning (G51.2, G50.2)

Command

**G51.2** P_ Q_

switches polygonal turning mode on. The revolution ratio of the workpiece and the tool can be set at addresses P and Q. If, e.g., the workpiece and the tool has to rotate 1:2, G51.2 P1 Q2 is to be programmed. The value range of addresses P and Q:

P = 1...127

Q = –127...+127

also a negative number can be set at address Q, this results in counter-direction of the tool spindle rotation.

Command

**G50.2**

switches polygonal turning off.

Commands G51.2 and G50.2 must always be issued in separate blocks.

In order to execute polygonal turning, a second spindle, rotating the tool, must also be mounted on the machine tool. Both spindles, i.e. the work spindle and the tool spindle must be equipped with encoders. When switching the mode on, first the NC speeds the tool spindle up to the revolution resulting from the values programmed to work spindle (address S) and to address P, Q: $n_{toolspindle} = S \cdot (Q/P)$. Afterwards the zero pulses of the two encoders are synchronized, than the tool spindle is rotated on the basis of the movement measured on the work spindle encoder as to the programmed Q/P ratio.

The effect of synchronizing is similar to that of the spindle zero pulse in case of thread cutting. Till the work is not removed from the chuck and the angle of tool is kept unchanged in tool spindle, the movement of workpiece and tool remains synchronized, the same surface can be machined more times, e.g. for the sake of roughing and finishing. The synchronizing can also be switched off by means of command G50.2, than by re-switching it the surface of the polygon turned previously can be turned again, provided the programmed revolution coincides with the previous one.

☞ *Warning!*

*When programming polygonal turning, take care, that the tool spindle revolution $n_{toolspindle} = S \cdot (Q/P)$ never exceeds the enabled maximum revolution.*

Synchronized run is switched off by

– emergency stop,

– servo errors.

Example

```
...
G0 X120 Z5 T505
S1000 M3                      (start work spindle 1000 mpr)
G51.2 P1 Q2                   (polygonal turning on, tool spindle rev 2000
                              m.p.r.)
G0 X100                       (cut in direction X)
G1 Z-50 F0.01                 (turn)
G4 P2                         (wait)
G0 X120                       (retract in X)
G50.2                         (polygonal turning off)
...
```

# 20 Measurement Functions

## 20.1 Skip Function (G31)

Instruction

**G31** v (F) (P)

starts linear interpolation to the point of v coordinate. The motion is carried on until an external skip signal (e.g. that of a touch-probe) arrives or the control reaches the end-point position specified at the coordinates of v. The control will slow down and come to a halt after the skip signal has arrived.

Address P specifies which skip signal input is to be used during movement of the 4 ones available in control:

**P0**: Uses skip signal 1
**P1**: Uses skip signal 2
**P2**: Uses skip signal 3
**P3**: Uses skip signal 4

If address P is not specified, control takes skip signal 1.

G31 is a non-modal instruction applicable only in the particular block, in which it has been programmed. The control returns error message *3051 G22, G28, ... G31, G37* if a syntactic error is found in instruction G31.

The speed of motion is
– the specified or modal value F if parameter *SKIPF*=0
– the feed value taken from *G31FD* if parameter *SKIPF*=1.

In the instant the external signal arrives, the positions of axes will be stored in the system variables specified below.

#5061.........position of axis 1
#5062.........position of axis 2
.
.
#5068.........position of axis 8

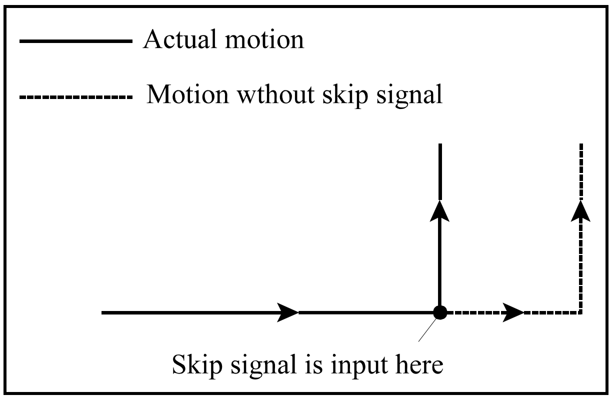


Fig. **20.1-1**

The position stored there is
– the position assumed in the instant the external signal (if any) has arrived,
– the programmed end-point position of interpolation G31 (unless an external signal has arrived),
– in the current work coordinate system,
– with the actual length compensation.

The motion comes to a halt with linear deceleration after the external signal has arrived. Now the end-point position of interpolation G31 is slightly different from the positions stored in variables #5061... on arrival of the signal, the difference varies with the feed applied in the interpolation. The end-point positions of the interpolations are accessible in variables #5001... . The next interpolation will be effective from those end-point positions on.

The interpolation can be executed in state G40 only. Programming G31 in state G41 or G42 returns error message *3054 G31 IN INCORRECT STATE*. Again, the same error message will be returned if state G95, G51, G51.1, G68 or G16 is in effect.

The value specified at coordinates v may be an incremental or an absolute one. If the next movement command following G31 block is specified in incremental coordinates, the motion will be calculated from the point where the skip signal has arrived and the motion stopped.

For example,

```
        N1 G31 G91 Z100
        N2 X100 Z30
```

An incremental motion in direction Z is started in block N1. If the control comes to a halt at the point of coordinate Z=86.7 on arrival of the external signal, it will move incrementally 100 in X direction and 30 in Z direction in block N2 (reckoned from that point).

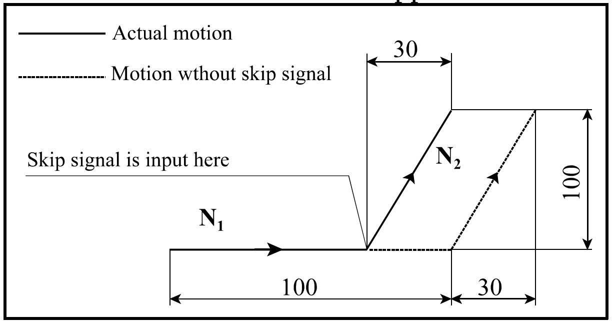


Fig. **20.1-2**

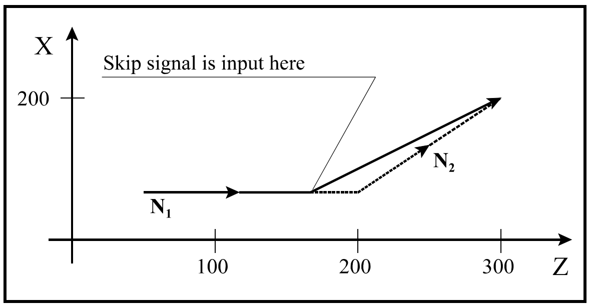


Fig. **20.1-3**

In the case of an absolute data specification being programmed, the motion will be

```
        N1 G31 G90 X200
        N2 X300 Z300
```

Interpolation N1 starts a motion in direction Z to the point of coordinate Z=200. If, after arrival of the external signal, the control comes to a halt at the point of coordinate Z=167, the displacement in direction X will be Z=300-167, i.e., Z=133 in block N2.

## 20.2 Automatic Tool Length Measurement (G36, G37)

Instruction

**G36** X__,

**G37** Z__

will cause the motion to be started in rapid traverse in the direction of X when G36 was specified and in the direction of Z when G37. The value of X or Z is interpreted invariably as an absolute data and it is the predicted position of the measuring sensor.

The motion will be carried on in rapid traverse rate as far as position X - *RAPDISTX*, or Z - *RAPDISTZ* where *RAPDISTX RAPDISTZ* are parameter-selected values.

The motion is then carried on with the feed specified in parameter *G37FD* until the signal of the probe arrives or until the control returns the error message *3103 OUT OF RANGE*. The latter occurs only when the touch-probe signal

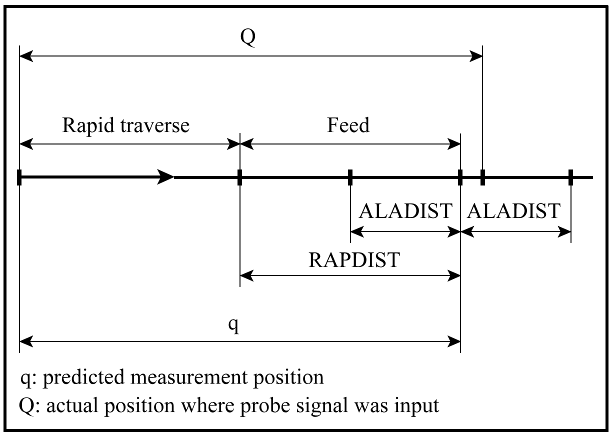

Q
Rapid traverse
Feed
ALADIST
ALADIST
RAPDIST
q
q: predicted measurement position
Q: actual position where probe signal was input

Fig. **20.2-1**

arrives outside of the ALADIST range (specified on parameter) of the predicted position X or Z.
If the measurement is completed successfully and the touch-probe signal has arrived at the point of coordinate Q, the control will

– add the difference Q-q to the wear compensation register selected earlier (if parameter *ADD*=1)
– or will subtract the difference from it (if parameter *ADD*=0).

The appropriate length compensation register have to be set up by programing Tnnmm prior to commencement of the measurement.

– G36, G37 are single-shot instructions.
– Cycle G36, G37 will be executed invariably in the coordinate system of the current workpiece.
– Parameters *RAPDIST* and *ALADIST* are always positive values. The condition *RAPDIST* > *ALADIST* must be fulfilled for the two parameters.
– Error message *3051 G22, G28, ... G31, G37* will be returned in the case of a syntactic error.
– Code Tnnmm referring to a length compensation cannot be specified in block G37, or else error message *3055 G37 IN INCORRECT STATE* is returned.
– Again, the same error message is returned when state G51, G51.1, G68 is in effect.

The following error message will be returned during the execution of function G36 or G37.

– Message *3103 OUT OF RANGE* is returned if the touch-probe signal arrives outside of the *ALADISTX* or *ALADISTZ* range of the end position programmed in interpolation G36, G37.

# 21 Safety Functions

## 21.1 Programmable Stroke Check (G22, G23)

Instruction

**G22** X Y Z I J K P

will forbid to enter the area selected by the command. Meaning of addresses:

| | |
|---|---|
| X: | Limit along axis X in positive direction |
| I: | Limit along axis X in negative direction |
| Y: | Limit along axis Y in positive direction |
| J: | Limit along axis Y in negative direction |
| Z: | Limit along axis Z in positive direction |
| K: | Limit along axis Z in negative direction |

The following conditions must be fulfilled for the specified data:

$$X \geq I, \quad Y \geq J, \quad Z \geq K$$

It can be selected at address P that the area is prohibited on the outside or on the inside.
P=0, the selected area is prohibited on the inside.
P=1, the selected area is prohibited on the outside.

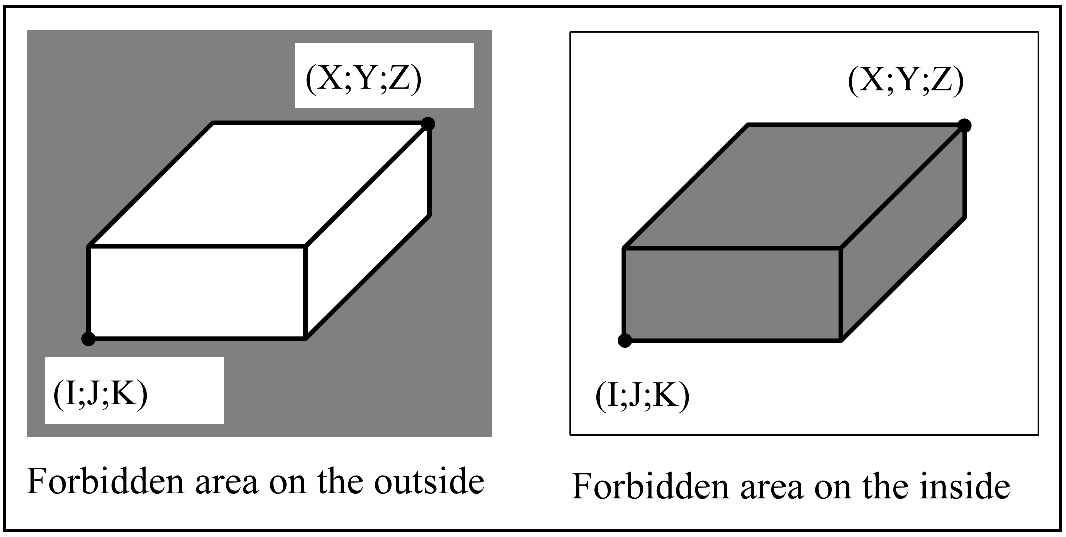


Fig. **21.1-1**

Instruction

**G23**

will cancel programmable stroke check function, the tool can enter the area selected above.
Instruction G22, G23 will re-write directly the respective parameters.
Instruction G22 or G23 will set parameter *STRKEN* to 1 or 0, respectively.
Instruction G22 P0 or G22 P1 will set parameter *EXTER* to 0 or 1, respectively.
Coordinates X, Y, Z in instruction G22 will write the *LIMP2n* parameters pertaining to the respective axes, coordinates I, J, K will set the *LIMN2n* values pertaining to the respective axes. Before being written to the respective parameters, the coordinates in instruction G22 will be converted to the coordinate system of the machine, with the selected compensation offsets included. Thus e.g., if the length compensation is set up in direction Z when instruction G22 is specified, the limit data of coordinates specified for that axis will limit the movement by stopping the tip of the tool at the limit. If, however, the compensation is not set up, the reference point of the tool

holder will not be allowed into the prohibited area. It is advisable to set the border of the forbidden area at the axis of the tool for the longest one.

– Programable stroke check function is not available for the additional axes.
– Instructions G22, G23 have to be specified in independent blocks.
– Programable stroke check function will be effective after reference point return.
– If the machine enters a prohibited area after reference-point return or as a result of programming G22, and the area is prohibited internally, the prohibition has to be released in manual mode by programming G23; the axis/axes must be moved out by manual jog, and stroke check has to be set up again by programming G22. In the case of an externally forbidden area, the procedure of leaving the area will be the same as the one following an overtravel.
– If an axis reaches the border of the prohibited area in motion, it can be removed from it by manual movement (in one of the manual modes).
– The entire space is allowed if X=I, Y=J, Z=K and E=0.
– The entire space is prohibited if X=I, Y=J, Z=K and E=1.
– If the area is prohibited internally, and the axes reach the prohibited area or a border thereof, the control will return the error message *1400 INTERNALLY FORBIDDEN AREA*.
– If the area is prohibited externally, and the axes reach the prohibited area or a border thereof, the control will return the error message *FORBIDDEN AREA t+* or *FORBIDDEN AREA t–* where t is the name of axis.

## 21.2 Parametric Overtravel Positions

Using the parameters of the control, the machine-tool builder can define for each axis the overtravel positions that is the stroke limit permissible with the particular machine. As soon as the border of that area is reached, the control will return an error message as if it had run over a limit switch.

- Parametric overtravel function is only performed by the control after reference point has been returned.
- The parametric overtravel function will prohibit always an external area.
- The areas of programmed stroke check and that of overtravel functions may overlap.

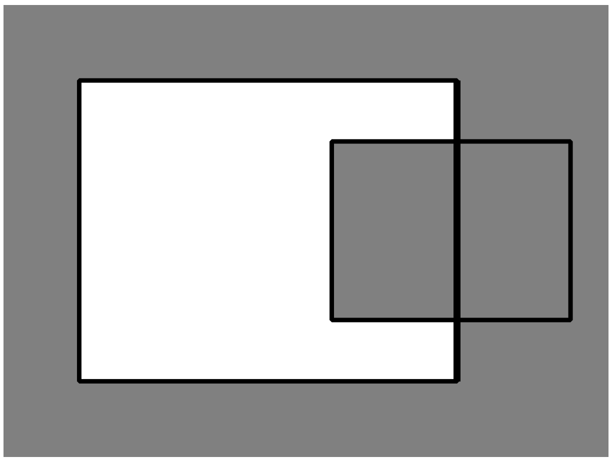
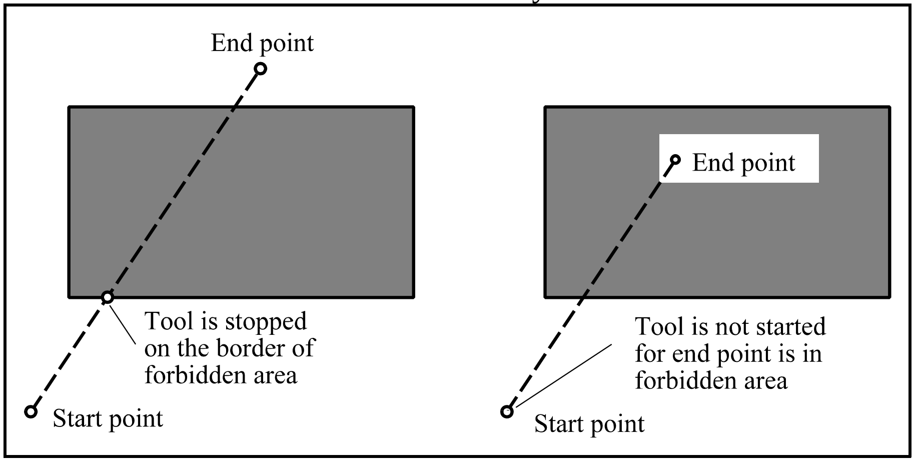


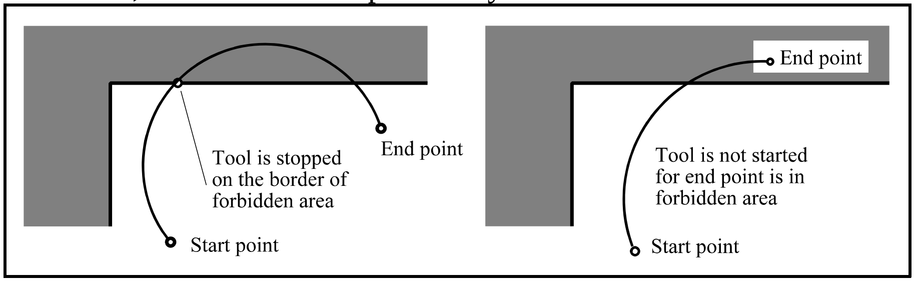
Fig. **21.2-1**

## 21.3 Stroke Check Before Movement

The control differentiates two forbidden areas. The first is the parametric overtravel area which delimits the physically possible movement range of the machine. The extreme positions of that range are referred to as limit positions. During movements the control will not allow those axes to move beyond the limits of that area defined by parameters. The limit positions are set by the builder of the machine, the user may not alter those parameters.
The second is the area defined by the programmable stroke check function. This may be accomplished by programming command G22 or rewriting the parameters.
During any motion the control will not allow the axes to move beyond the limits of these areas.
If parameter *CHBFMOVE* is set to 1, the control will - before starting the axes in the course of executing a block - check whether the programmed end point of the particular interpolation is in a prohibited area.
If the end point of the block is located outside of the parametric overtravel area or in the programmed forbidden area, error message *3056 LIMIT* or *3057 FORBIDDEN AREA* will be returned, respectively. As a result, the movement is practically not started at all.


End point
Tool is stopped on the border of forbidden area
Start point
End point
Tool is not started for end point is in forbidden area
Start point
Fig. 21.3-1

Since, prior to starting the interpolation, the control only checks whether the end point of the interpolation is located in a prohibited area the error message is produced in the instances shown in the figures at the border of the forbidden area, after the movement has been started.


Tool is stopped on the border of forbidden area
End point
Start point
End point
Tool is not started for end point is in forbidden area
Start point
Fig. 21.3-2

# 22 Custom Macro

## 22.1 The Simple Macro Call (G65)

As a result of instruction

**G65** P(program number) L(number of repetitions) <argument assignment>

the custom macro body (program) specified at address P (program number) will be called as many times as is the number specified at address L.
Arguments can be assigned to the macro body. They are specific numerical values assigned to definite addresses, that are stored in respective local variables during a macro call. Those local variables can be used by the macro body, i.e., the macro call is a special subprogram call in which the main program can transfer values (parameters) to the subprogram.
The following two argument assignments can be selected:

Address string of argument assignment No.1 is

**A B C D E F H I J K M Q R S T U V W X Y Z**
No value can be transferred to the macro body at any one of addresses **G, L, N, O, P**. The addresses can be filled in any arbitrary sequence, not necessarily in alphabetical order.

Address string of selecting argument assignment No.2 is

**A B C I1 J1 K1 I2 J2 K2 ... I10 J10 K10**
In addition to addresses A, B, C, maximum 10 different arguments can be assigned for addresses I, J, K. The addresses can be filled in any arbitrary sequence. If several arguments are selected for a particular address, the variables will assume the respective values in the order of selection.

| *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 |

- **Abbreviations**: *lv*=local variable, *1.a a*=argument assignment No.1, *2.a a*= argument assignment No.2.

The subscripts following addresses I, J, K indicate the argument assignment sequence.

The control will accept simultaneous selections of arguments 1 and 2 in a given block. An error message will be returned when an attempt is made to make reference twice to a variable of a particular number. For example,

```
             G65 A2.12 B3.213 J36.9 J–12 E129.73 P2200

Variable
#1=2.12
#2=3.213
#5=36.9
#8=–12
#8=Error
```

In the above example, variable #8 has already been assigned a value by the second address J (value, -12), since the value of address E is also assigned to variable #8, the control returns error message *3064 BAD MACRO STATEMENT.*
A decimal point and a sign can also be transferred at the addresses.

## 22.2 The Macro Modal Call

### 22.2.1 Macro Modal Call in Every Motion Command (G66)

As a result of instruction

**G66** P(program number) L(number of repetitions) <argument assignment>

the macro body specified at address P (program number) will be called after the execution of each motion command, as many times as is the number specified at address L. The interpretations of addresses P and L and the rules of argument assignment are identical with those described for instruction G65.
The selected macro will be called until with command

**G67**

it is canceled.

For example, a hole has to be drilled in a given segment of the part program after each movement:

Main program

```
  ...
  G66 P1250 Z–100 R–1 X2 F130    (Z=Z point of hole, R=R point of hole,
                                 X=dwell F=feed)
  G91 G0 X100
  Y30                            drilling is performed after each posi-
                                 tioning
  X150
  ...
  G67
```

Macro body

```
 %O1250
 G0 Z#18              (rapid-traverse positioning in direction Z to the
                      point specified at address R–1)
 G1 Z#26 F#9          (drilling as far as the point Z specified at address
                      Z–100, with the feed specified at address F130)
 G4 P#24              (dwell at the bottom of the hole for the time
                      specified at address X2)
 G0 Z-[#18+#26]       (retraction of the tool to the initial point)
 M99                  (return to the main program)
```

```
%
```

## 22.2.2 Macro Modal Call From Each Block (G66.1)

As a result of command

**G66.1** P(program number) L(number of repetitions) <argument assignment>

all subsequent blocks will be interpreted as argument assignment, and the macro of the number specified at address P will be called, and will be executed as many times as is the number specified at address L.

The command produces the same effect as if each block were a G65 macro call:

```
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
```

The selected macro will be called until with command

**G67**

it is canceled.

The rules of argument assignment are

1. In the block performing the activation (in which G66.1 P L has been programmed), the rules of the argument assignment are the same as in command G65.
2. In the blocks following instruction G66.1,
   the same addresses can be used as in command G65, and
   L: #12,
   P: #16,
   G: #10 with the qualification that the control will accept only one reference to an address G in each block; programming several G addresses will produce error message *3005 ILLEGAL G CODE*.
   N: #14 if an N address is at the beginning of a block (or preceded at most by the address of a conditional block "/"), the second N address will be considered for an argument:

```
                 /N130 X12.3 Y32.6 N250
                         │     │     │     │
Block No.  ──────────────┘     │     │     │
#24=12.3  ─────────────────────┘     │     │
#25=32.6  ───────────────────────────┘     │
#14=250   ─────────────────────────────────┘
```

   If address N is in the middle of the block (preceded by any address other than "/"), address N will be interpreted as an argument:

```
                 X34.236  N320
                        │        │
#24=34.236  ────────────┘        │
#14=320  ────────────────────────┘
```

   if address N has been recorded already as an argument, the next reference to address N will produce error message *3064 BAD MACRO STATEMENT.*

In the case of G66.1, the rules of block execution:

The selected macro will be called already from the block, in which code G66.1 has been specified, taking into account the rules of argument assignment described at point 1.

Each NC block following G66.1 to a block containing code G67 will produce a macro call with the rules of argument assignment described under point 2. No macro will be called if an empty block is found (e.g., N1240) where a reference is made to a single N address, or from a block containing a macro instruction.

## 22.3 Custom Macro Call Using G Code

Maximum 10 different G codes can be selected by parameters, to which macro calls are initiated. Now instead of specifying

Nn G65 Pp <argument assignment>

the following command can be used

Nn **Gg** <argument assignment>.

The particular program number to be called by the G code has to be selected in parameters. None of codes G65, G66, G66.1 and G67 may be specified for this purpose.

G(9010)=code G calling program O9010
G(9011)=code G calling program O9011
:
G(9019)=code G calling program O9019

If a negative value is written in parameters, the selected G code will generate a modal call. If, e.g., G(9011)=-120, instruction G120 in the program will produce a modal call. The state of parameter *MODGEQU* will define the type of call:

MODGEQU=0, call is G66 type
MODGEQU=1, call is G66.1 type.

If the value of the parameter is 0, the macro will be called at the end of each motion block. If the value of the parameter is 1, the macro will be called for each block.
If a standard G code is selected for user call (e.g., G01), and a reference is made to that code again in the body of the macro, it will not produce another call, instead, it will be interpreted and executed by the control as an ordinary G code.
If a reference is made to the calling G code again in the body of the macro, and it is different from a standard G code, the control will return error message *3005 ILLEGAL G CODE*.

– Calling a user M, S, T, A, B, C from a user G code call,
– Calling a user G code from a user M, S, T, A, B, C call is enabled, depending on the parameter value.

FGMAC=0, not enabled (executed as ordinary M, S, ... G codes)
FGMAC=1, enabled, i.e. a new call is generated.

The user G codes have the following sets of arguments:

– If the code is of G65 or G66 type, the set of arguments assigned to G65, plus P and L,
– if the code is of G66.1 type, the points described are applicable to its set of arguments.

A modal code can be deleted by instruction G67.

## 22.4 Custom Macro Call Using M Code

Maximum 10 different M codes can be selected by parameters, to which macro calls are initiated. Now the series of instructions

Nn **Mm** <argument assignment>

have to be typed. Now code M will not be transferred to the PLC, but the macro of the respective program number will be called.

The particular program number to be called by the calling M code has to be selected by parameters.

M(9020)=code M calling program O9020
M(9021)=code M calling program O9021
:
M(9029)=code M calling program O9029

Code M can specify invariably a type G65 call (i.e., a non-modal one).
If reference is made again to the same M code in the middle of the macro body, the latter will not call the macro, instead, M code will be transferred to the PLC.
If a user call type G, S, T, A, B, C or some other user call type M is made in the middle of the macro body,

FGMAC=0, not enabled (executed as ordinary M, S, ... G codes)
FGMAC=1, enabled, i.e. a new call is generated.

An M code selected by parameters to initiate a macro call may be preceded only by "/" and address N in the block.
A block containing a macro call initiated by M code may include a single M code only.
Set of arguments No.1:

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

Set of arguments No. 2 also can be used with function M.

## 22.5 Subprogram Call with M Code

Maximum 10 M codes can be selected by parameters, by which subprogram calls can be initiated.
Now instead of instruction

Nn Gg Xx Yy M98 Pp

can be specified

Nn Gg Xx Yy **Mm**

Now the selected M code will not be transferred to the PLC, instead, the respective subprogram will be called.
The particular program number to be called by M code can be selected by the following parameters.

M(9000)=code M calling program O9000
M(9001)=code M calling program O9001
:
M(9009)=code M calling program O9009

If reference is made to the same M code again in the subprogram, the latter will not call the subprogram again, but M code will be transferred to the PLC.
If a user call G, S, T, A, B, C or some other user call M is made in the subprogram:
FGMAC=0, not enabled (executed as an ordinary codes M, S, ... G)
FGMAC=1, enabled, i.e. a new call will be generated.

## 22.6 Subprogram Call with T Code

With parameter T(9034)=1 set, the value of T written in the program will not be transferred to the PLC, instead, the T code will initiate the call of subprogram No. O9034.
Now block

Gg Xx Yy **Tt**

will be equivalent to the following two blocks:

**#199=t**
Gg Xx Yy M98 P9034

The value assigned to address T will be transferred as an argument to common variable #199.
If reference is made to address T again in the subprogram started upon code T, the subprogram will not be called over again, but the value of address T will be transferred already to the PLC.
If a user call of G, M, S, A, B, C is made in the subprogram,
FGMAC=0, not enabled (executed as an ordinary codes M, S, ... G)
FGMAC=1, enabled, i.e. a new call is generated.

## 22.7 Subprogram Call with S Code

With parameter S(9033)=1 set, the value of S written in the program will not be transferred to the PLC, instead, the call of subprogram O9033 will be initiated by the S code.
Now block

Gg Xx Yy **Ss**

is equivalent to the following two blocks:

**#198=s**
Gg Xx Yy M98 P9033

The value assigned to address S will be transferred as an argument to common variable #198.
If reference is made to address S again in the subprogram started by S code, the subprogram will not be called again, but the value of the address will be transferred already to the PLC.
If a user call of G, M, T, A, B, C is made in the subprogram,
FGMAC=0, not enabled (executed as an ordinary codes M, S, ... G)
FGMAC=1, enabled, i.e. a new call is generated.

## 22.8 Subprogram Call with A, B, C Codes

If address A, B or C is defined as an auxiliary function by parameters (1493 A.MISCEL=1, 1496 B.MISCEL=1, or 1499 C.MISCEL=1) and parameter A(9030)=1, or B(9031)=1, or C(9032)=1 is set, the value of A, B or C written in the program will not be transferred to the PLC or the interpolator, instead the call of subprogram No.O9030, O9031 or O9032 will be initiated by code A, B or C, respectively.
Now e.g. block

Gg Xx Yy **Bb**

is equivalent to the following two blocks:

**#196=b**
Gg Xx Yy M98 P9031

The values assigned to addresses **A**, **B** and **C** will be transferred to common variables **#195**, **#196**,and **#197**, respectively.
If reference is made again to the same address in the subprogram started by code A, B or C, the subprogram will not be called again, but the value of the address will be transferred already to the PLC or interpolator.

If a call of a user G, M, S, T code is made in the subprogram,
 FGMAC=0, not enabled (executed as ordinary codes M, S, ... G)
 FGMAC=1, enabled, i.e. a new call is generated.

## 22.9 Differences Between the Call of a Subprogram and the Call of a Macro

– A macro call may include arguments, but a subprogram call may not.
– The call of a subprogram will only branch into the subprogram after the execution of other commands programmed in the block; a macro call will branch only.
– A macro call will alter the levels of local variables, a subprogram call will not. For example, the value of #1 prior to the call of G65 is different from the one in the middle of macro body. The value of #1 before M98 is identical with that in the subprogram.

### 22.9.1 Multiple Calls

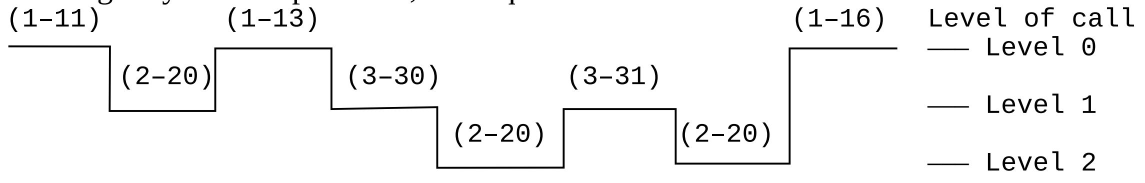
Another macro can be called again from a macro. Macro calls can be made in four levels of depth, including simple and modal ones. With the subprogram calls included, the maximum depth of the calls may cover 8 levels.
In the case of multiple calls of modal macros (type G66), first the latter specified macro will be called after execution of each interpolation block, from which the previously specified macros will be called in a backward sequence. 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            (canceling of call G66 P3...)
N15 G67            (canceling of call G66 P2 ...)
N16 Z–5            (1–16)
...

%O0002
N20 X4             (2–20)
N21 M99
%

%O0003
N30 Z2             (3–30)
N31 Z3             (3–31)
N32 M99
%
```

Including only the interpolations, the sequence of execution will be



Of the numbers in brackets, the first and the second ones are the numbers of the programs and block being executed, respectively.

Instruction G67 specified in block N14 will cancel the macro called in block N12 (O0003); the one specified in block N15 will cancel the macro called in block N10 (O0002).
In the case of multiple calls of macros type G66.1, first the last specified macro will be called in entering each block (treating the addresses of the particular block as arguments), then the previously specified macro will be called, entering the blocks of that macro and treating them as arguments.
If another macro is called again from a macro, the levels of local variables will also increase with the macro levels.

```
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 will be stored (#1 through #33), and the local variables at level 1 will assume the argument values specified in the call. If another macro is called from the first level, the local variables of the first level will be stored (#1 through #33), and the local variables on the second level will assume the argument values specified in the call. In the case of multiple call, the local variables of the previous level will be stored and the local variables on the next level will assume the argument values specified in the call. In the case of M99, returning from the called macro to the calling program, the local variables stored on the previous level will be restored in the same states they were at the time of being stored during the call.

## 22.10 Format of Custom Macro Body

The program format of a user macro is identical with that of a subprogram:

```
O(program number)
:
commands
:
M99
```

The program number is irrelevant, but the program numbers between O9000 and O9034 are reversed for special calls.

## 22.11 Variables of the Programming Language

Variables instead of specific numerical values can be assigned to the addresses in the main programs, subprograms and macros. A value can be assigned to each variable within the permissible range. The use of variables will make for much more flexible procedures of programming.
The appropriate data can be parametrized by the use of common variables in the main programs and subprograms, thus it will not be necessary to write new programs for similar work parts of different size. Instead, the operator can change to new part of different size by re-writing the appropriate common variables.
The use of variables can make a macro much more flexible than a conventional subprogram. Whereas arguments cannot be transferred to a subprogram, arguments can be attached to a macro through the local variables.

### 22.11.1 Identification of a Variable

A number of variables can be used, and each will be identified by its number. A variable is composed of the code # and a number. For example,

```
#12
#138
#5106
```

A formula may also be used to make reference to variable - #[<formula>]
For example,

#[#120] means that variable 120 contains the serial number of variable that is referred to;
#[#120-4] means that the referred variable number is obtained by subtracting 4 from the number contained in the variable.

### 22.11.2 Referring to a Variable

The various addresses in the words of a program block can assume values of variables as well as numerical values. The minus sign ("–") or operator I can, wherever it is permissible with numerical values, be used even when a reference is made to a variable after an address. 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
```

– Referring to program number O, block number N or conditional block / by a variable is not permissible. Address N will be regarded as a block number if it is preceded only by address "/" in the block.
– The number of a variable may not be substituted for by a variable, i.e. ##120 is not permissible. The correct specification is #[#120].
– If the variable is used behind an address, its value may not exceed the range of values permissible for the particular address. If, e.g., #112=5630, reference M#112 will produce an error message.
– If the variable is used behind an address, its value will be rounded to a significant digit corresponding to the address. For example,

```
M#112      will be M1    for    #112=1.23
M#112      will be M2    for    #112=1.6
```

### 22.11.3 Vacant Variables

A variable that has not been referred to (undefined) is vacant. Variable #0 is used for a variable that is always vacant:

#0=<vacant>

### 22.11.4 Numerical Format of Variables

Each variable is represented by 32 bits of mantissa and 8 bits of characteristic,

variable= $M*2^C$

| | |
|---|---|
| Representation of a **vacant** variable, | M=0, C=0 |
| Representation of a **0 - value** variable, | M=0, C=–128 |

The nature of a vacant variable, compared in an address:

*Reference to* a **vacant** variable in an address:

```
If #1=<vacant>         if #1=0

G90 X20 Y#1         G90 X20 Y#1
     |                   |
  G90 X20           G90 X20 Y0
```

**Vacant** variable in a *definition* instruction:

```
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
```

Difference between a **vacant** variable and a **0 - value** one in a **conditional expression** will be

```
if #1=<vacant>         if #1=0

 #1 EQ #0            #1 EQ #0
     |                  |
 fulfilled         not fulfilled

 #1 NE 0             #1 NE 0
     |                  |
 fulfilled         not fulfilled

 #1 GE #0            #1 GE #0
     |                  |
 fulfilled         not fulfilled

 #1 GT 0             #1 GT 0
     |                  |
 fulfilled         not fulfilled
```

### 22.12 Types of Variables

With reference to the ways of their uses and their properties, the variables are classified into local, common and system variables. The number of the variables tells the particular category to which it pertains.

#### 22.12.1 Local Variables (#1 through #33)

The local variable is a variable used by the macro program locally. If macro A calls B, and reference is made to local variable #i in each of macros A and B, the value of local variable #i at the level macro A will not be lost and will not be re-written after macro B has been called - despite the fact that reference is made to #i in macro B as well. The local variables are used for the transfer of arguments. The matches between the addresses of arguments and the local variables are contained in the Table in the Section describing the procedure of a simple macro call (G65).
The local variable whose address has not been involved in the argument assignment, is a vacant one that can be used optionally.

#### 22.12.2 Common Variables (#100 through #199, #500 through #599)

Unlike the local variables, the common variables are identical throughout the entire program (not only at the given levels of program calls) - regardless of whether they are in the main program, a subprogram or in a macro, or at whatever level of the macro. If accordingly, #i has been used in a macro, e.g. a value has been assigned to it, #i will have the same value in another macro, too, until it is re-written. The common variables can be used absolutely freely in the system, they have no distinguished functions at all.

The common variables from #100 to #199 will be deleted upon a power-off.
The values of common variables #500 through #599 will be preserved even after a power-off.
The macro variables #500 through #599 can be made "write-protected" by the use of parameters *WRPROT1* and *WRPROT2*. The number of the first and the last element of the block to be protected will be written to parameters *WRPROT1* and *WRPROT2*, respectively. If, e.g., the variables #530 through #540 are to be protected, the respective parameters have to be set as *WRPROT1*=530 and *WRPROT2*=540.

### 22.12.3 System Variables

The system variables are fixed ones providing information about the states of the system.

**Interface input signals - #1000–#1015, #1032**

16 interface input signals can be determined, one by one, by reading the system variables #1000 through #1015.

| Name of system variables | Interface input with reference to the PLC program |
|---|---|
| #1000 | I[CONST+000] |
| #1001 | I[CONST+001] |
| #1002 | I[CONST+002] |
| #1003 | I[CONST+003] |
| #1004 | I[CONST+004] |
| #1005 | I[CONST+005] |
| #1006 | I[CONST+006] |
| #1007 | I[CONST+007] |
| #1008 | I[CONST+010] |
| #1009 | I[CONST+011] |
| #1010 | I[CONST+012] |
| #1011 | I[CONST+013] |
| #1012 | I[CONST+014] |
| #1013 | I[CONST+015] |
| #1014 | I[CONST+016] |
| #1015 | I[CONST+017] |

where CONST=I_LINE*10 and *I_LINE* is a parameter. Thus, any arbitrary interface input can be read.
The values of the above variables are

0= if the contact at the input is open,

1= if the contact at the input is closed.

The above 16 inputs can be read simultaneously at variable #1032. Depending on the system variables assigned to the one-by-one reading, the value will be

$$\#1032=\sum_{i=0}^{15}\#[1000+i]*2^{i}$$

Accordingly, with 24V applied to inputs #1002 and #1010, the rest of inputs being open, the value of variable #1032 will be

$$\#1032 = 1*2^{2} + 1*2^{10} = 1028$$

The variables of the interface inputs are "read only" ones, and may not be used on the left side of a definition instruction.

## Interface output signals - #1100–#1115, #1132

16 interface output signals can be issued, one by one, by assigning values to variables #1100 through #1115.

| Name of system variables | Interface input with reference to the PLC program |
|---|---|
| #1100 | Y[CONST+000] |
| #1101 | Y[CONST+001] |
| #1102 | Y[CONST+002] |
| #1103 | Y[CONST+003] |
| #1104 | Y[CONST+004] |
| #1105 | Y[CONST+005] |
| #1106 | Y[CONST+006] |
| #1107 | Y[CONST+007] |
| #1108 | Y[CONST+010] |
| #1109 | Y[CONST+011] |
| #1110 | Y[CONST+012] |
| #1111 | Y[CONST+013] |
| #1112 | Y[CONST+014] |
| #1113 | Y[CONST+015] |
| #1114 | Y[CONST+016] |
| #1115 | Y[CONST+017] |

where CONST=O_LINE*10 and *O_LINE* is a parameter. Thus, any arbitrary interface output word can be issued or read.
The values of the above variables may be

0= the contact at the output is open,

1= the contact at the output is closed.

The above 16 outputs can be issued simultaneously by using variable #1132. Depending on the system variables assigned to the single outputs, the output value will be

$$\#1132=\sum_{i=0}^{15}\#[1100+i]*2^{i}$$

Accordingly, with outputs #1102 and #1109 are on, the rest of outputs being off, variable #1132 must output the value

$$\#1132 = 1*2^{2} + 1*2^{9} = 516$$

## Tool compensation values - #10001 through #13999

The tool compensation values can be read from variables #10001 through #19999, or values can be assigned them.

| N | X | | Y | | Z | | R | | Q |
|---|---|---|---|---|---|---|---|---|---|
| | wear | geom.. | wear | geom.. | wear | geom.. | wear | geom.. | |
| 1 | #10001 | #15001 | #14001 | #19001 | #11001 | #16001 | #12001 | #17001 | #13001 |
| 2 | #10002 | #15002 | #14002 | #19002 | #11002 | #16002 | #12002 | #17002 | #13002 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 99 | #10099 | #15099 | #14099 | #19099 | #11099 | #16099 | #12099 | #17099 | #13099 |

## Work zero-point offsets - #5201 through #5328

The work zero-point offsets can be read at variables #5201 through #5328, or values can be assigned them.

| No. of variable | value of variable | workpiece coordinate system |
|---|---|---|
| #5201 | common work zero point offset, axis 1 | common for all the coordinate systems |
| #5202 | common work zero point offset, axis 2 | |
| : | | |
| #5208 | common work zero point offset, axis 8 | |
| #5221 | work zero point offset value, axis 1 | G54 |
| #5222 | work zero point offset value, axis 2 | |
| : | | |
| #5228 | work zero point offset value, axis 8 | |
| #5241 | work zero point offset value, axis 1 | G55 |
| #5242 | work zero point offset value, axis 2 | |
| : | | |
| #5248 | work zero point offset value, axis 8 | |
| #5261 | work zero point offset value, axis 1 | G56 |
| #5262 | work zero point offset value, axis 2 | |
| : | | |
| #5268 | work zero point offset value, axis 8 | |
| #5281 | work zero point offset value, axis 1 | G57 |
| #5282 | work zero point offset value, axis 2 | |
| : | | |
| #5288 | work zero point offset value, axis 8 | |
| #5301 | work zero point offset value, axis 1 | G58 |
| #5302 | work zero point offset value, axis 2 | |
| : | | |
| #5308 | work zero point offset value, axis 8 | |
| #5321 | work zero point offset value, axis 1 | G59 |
| #5322 | work zero point offset value, axis 2 | |
| : | | |
| #5368 | work zero point offset value, axis 8 | |

The axis number refers to the physical ones. The relationship between the numbers and the names of axes will be defined by the machine tool builder by parameters in group *AXIS*. Usually axes

1, 2 and 3 are assigned to addresses X, Y and Z, respectively, but different specifications are also permissible.

**Alarm - #3000**

By defining

#3000=nnn(ALARM),

a numerical error message (nnn=max. three decimal digits) and the text of error message can be provided. The text must be put in (,) brackets. A message may not be longer than 25 characters. If the macro contains an error, i.e., the program runs to a branch in which a value has been defined to variable #3000, the program will be executed as far as the previous block, then the execution is suspended and the error message and the code of it (4nnn) are displayed on the screen. The number of the message is the sum of number specified on #3000 variable and 4000. If no number was specified, the code of the message would be 4000 if no text was specified the message field will be empty. The error state can be canceled by the RESET button.

**Millisecond timer - #3001**

The value of variable #3001 can be read and written.
The time interval between two time instants can be measured in milliseconds, with an accuracy of about 20 ms. Counter #3001 will overflow at 65536. The value of variable #3001 will start from zero at the time of power-on, and will count upwards. Counting is continuous as long as the control is on.

**Main time timer - #3002**

The value of variable #3002 can be read and written.
The time interval between two time instants can be measured in minutes, with an accuracy of about 20 ms.
At the time of power-on, the value of variable #3002 will start at the power-off level and will be counted upwards.
Counting is on as long as the START light is on, i.e., the time is being measured in the start condition of the control. It is located at time meter *CUTTING2* of the parameter memory.

**Suppression of block-by-block execution - #3003**

If #3003=1, the control will not stop on completion of a block (in the state of single block mode) until that variable assumes value 0.
The value of the variable is 0 at power-off or after resetting the program to its beginning.

#3003 block-by-block execution
0 = not suppressed
1 = suppressed

## Suppression of stop button, feed override, exact stop - #3004

Under the conditions of suppression of feed stop function, the feed will stop after the stop button is pressed when the suppression is released.
When the feedrate override is suppressed, the override takes the value of 100% until the suppression is released.
Under the conditions of the suppression of the exact stop, the control will not perform a check until the suppression has been released.
The value of the variable is 0 at power-on or after resetting the program to its beginning.

| #3004 | Exact stop | Feed override | Feed 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 |

0 = function is effective
1 = function is suppressed

## Stop with message - #3006

As a result of a value assigned to

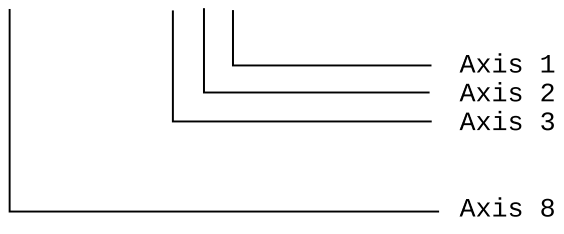
#3006=nnn(MESSAGE)

the execution of the program is stopped, and the message in round brackets and the code 5nnn will be displayed on the screen. The code is the sum of the number specified on the variable and 5000. If no number was specified, code 5000 would be displayed, if no text was specified message field would be empty. The execution of the program is resumed upon depression of the START button, then the message is cleared from the screen. The message may not be longer than 25 characters. This instruction is useful whenever the operator's intervention is needed during the execution of the program.

## Mirror image status - #3007

By reading variable #3007, the operator can establish the particular physical axis, on which mirror-image command is recorded. This variable is a "read only" one.
The value of the variable is interpreted in binary terms as follows.

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



The bits have the following meanings:

0 = no mirror imaging
1 = mirror imaging on.

If, e.g., the value of the variable is 5, mirror image is on in axes 1 and 3. The axis number refers to a physical axis, the parameter defining the particular name of axis pertaining to a physical axis number.

**Number of machined parts, number of parts to be machined - #3901, #3902**

The numbers of machined parts are collected in counter #3901 by the control. The contents of the counter will be incremented by 1 upon the execution of each function M02, M30 or selected M functions in parameter *PRTCNTM*. As soon as the number of machined parts becomes equal to the required number of parts (counter #3902), the NC tells it the PLC on a flag.

| | |
|---|---|
| Number of machined parts | #3901 |
| Number of parts to be machined | #3902 |

Counters #3901 and #3902 are located on parameters *PRTTOTAL* and *PRTREQRD*, respectively.

**Modal information - #4001 through #4130, #4201 through #4330**

The modal values effective in the previous block can be established by reading system variables #4001 through #4130.
The modal commands effective in the block under execution can be established by reading variables #4201 through 4330.

| system variable | modal information of the previous block | system variable | modal information of the block being executed |
|---|---|---|---|
| #4001 | G code, group 1 | #4201 | G code, group 1 |
| : | : | : | : |
| #4020 | G code, group 20 | #4220 | G code, group 20 |
| #4101 | code A | #4301 | code A |
| #4102 | code B | #4302 | code B |
| #4103 | code C | #4303 | code C |
| #4107 | code D | #4307 | code D |
| #4108 | code E | #4308 | code E |
| #4109 | code F | #4309 | code F |
| #4111 | code H | #4311 | code H |
| #4113 | code M entered first | #4313 | code M entered first |
| #4114 | block number, N | #4314 | block number, N |
| #4115 | program number, O | #4315 | program number, O |
| #4119 | code S | #4319 | code S |
| #4120 | code T | #4320 | code T |

## Positional information - #5001 through #5108

### Positions at block end

| system variable | position information | reading in during motion |
|---|---|---|
| #5001 | block end coordinate of axis 1 | |
| #5002 | block end coordinate of axis 2 | |
| : | | possible |
| #5008 | block end coordinate of axis 8 | |

The block end coordinate will be entered in the variable
– in the current work coordinate system
– with the coordinate offsets taken into account
– in Cartesian coordinates
– With all compensations (length, radius, tool offset) ignored.

### Instantaneous positions in the coordinate system of the machine

| system variable | nature of position information | entry during motion |
|---|---|---|
| #5021 | instantaneous coordinate of axis 1 (G53) | |
| #5022 | instantaneous coordinate of axis 2 (G53) | |
| : | | not possible |
| #5028 | instantaneous coordinate of axis 8 (G53) | |

The instantaneous position (G53) will be entered in the variable
– in machine coordinate system
– with all compensations (length, radius, tool offset) taken into account.

### Instantaneous positions in the work coordinate system

| system variable | nature of position information | entry during motion |
|---|---|---|
| #5041 | instantaneous coordinate of axis 1 | |
| #5042 | instantaneous coordinate of axis 2 | |
| : | | not possible |
| #5048 | instantaneous coordinate of axis 8 | |

The instantaneous coordinate of will be entered in the variable
– in the current work coordinate system
– with the coordinate offsets taken in account
– in Cartesian coordinates
– with all compensations (length, radius, tool offset) taken into account.

## Skip signal position

| system variable | nature of position information | entry during motion |
|---|---|---|
| #5061 | Skip signal coordinate of axis 1 (G31) | |
| #5062 | Skip signal coordinate of axis 2 (G31) | |
| : | | possible |
| #5068 | Skip signal coordinate of axis 8 (G31) | |

The position, in which the skip signal has arrived in block G31 will be entered in the variable
– in the work coordinate system
– with the coordinate offsets taken into account
– in Cartesian coordinates
– with all compensations (length, radius, tool offset) taken into account.
Unless the skip signal has arrived, the above variables will assume the end-point position programmed in block G31.

## Tool-length compensation

| system variable | nature of position information | entry during motion |
|---|---|---|
| #5081 | length compensation on axis 1 | |
| #5082 | length compensation on axis 2 | |
| : | | not possible |
| #5088 | length compensation on axis 8 | |

The readable tool-length compensation is the one in effect in the block being executed.

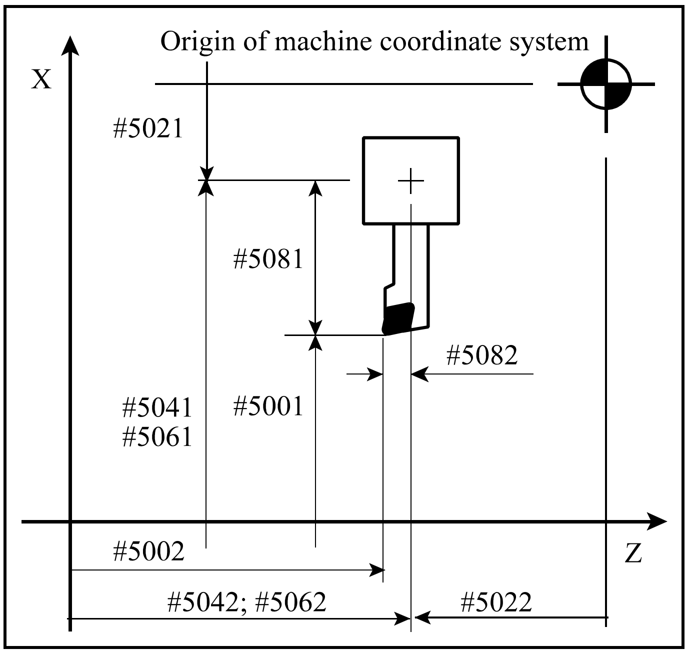


Fig. 22.12.3-1

**Servo lag**

```
system       nature of position information            entry during
variable                                                  motion

#5101        servo lag in axis 1
#5102        servo lag in axis 2
  :                                                    not possible
#5108        servo lag in axis 8
```

The readable servo lag is a signed value in millimeters.

## 22.13 Instructions of the Programming Language

The expression

#i = <formula>

is used for describing the various instructions. The expression <formula> may include arithmetic operations, functions, variables or constants.
In general, references are made to variables #j and #k in a <formula>.
It is not only possible for the <formula> to stand on the right side of a definition instruction, the various addresses in the NC block may also assume a formula instead of a specific numerical value or variable.

### 22.13.1 Definition, Substitution: #i = #j

The code of instruction is **=**.
As a result of the instruction, variable #i will assume the value of variable #j, i.e., the value of variable #j will be entered in variable #i.

### 22.13.2 Arithmetic Operations and Functions

***Single-Operand Operations***

**Single-operand minus:** #i = – #j

The code of the operation is **–**.
As a result of the operation, variable #i will have a value identical with variable #j in absolute value but opposite in sign.

**Arithmetic negation:** #i = NOT #j

The code of the operation is **NOT**
As a result of the operation, variable #j is converted first into a 32-bit fixed-point number. Error message *3091 ERRONEOUS OPERATION WITH #* is returned unless the converted number can be represented by 32 bits. Then the value of that fixed-point number will be negated bit by bit and the number produced this way will be re-converted into a floating-point one and will be put in variable #i.

***Additive arithmetic operations***

**Addition:** #i = #j + #k

The code of the operation is **+**.
As a result of the operation, variable #i will assume the sum of the values of variables #j and #k.

**Subtraction:** #i = #j – #k

The code of the operation is **–**.

As a result of the operation, variable #i will assume the difference of the values of variables #j and #k.

**Logical sum, or:** #i = #j OR #k

The code of the operation is **OR**.

As a result of operation, the logic sum of variables #j and #k will be entered in variable #i at every bit of 32 bits. Wherever 0 is found at each of the identical bit values of the two numbers, 0 will be represented by that bit value in the result (otherwise 1).

**Exclusive or:** #i = #j XOR #k

The code of the operation is **XOR**.

As a result of operation, the variables #j and #k will be added together in every bit of 32 bits in variable #i in such a way that 0 will be the bit value in the result wherever identical numerical values are found in identical bit positions (and 1 will be wherever different numerical values are found), in each of the 32 bits.

*Multiplicative arithmetic operations*

**Multiplication:** #i = #j * #k

The code of the operation is *****.

As a result of operation, variable #i will assume the product of the values of variables #j and #k.

**Division:** #i = #j / #k

The code of the operation is **/**.

As a result of operation, variable #i will assume the quotient of variables #j and #k. The value of #k may not be 0 or else the control will return error message *3092 DIVISION BY 0 #*.

**Remainder:** #i = #j MOD #k

The code of the operation is **MOD**.

As a result of the operation, variable #i will assume the remainder of the quotient of variables #j and #k. The value of #k may not be 0 or else the control will return error message *3092 DIVISION BY 0 #*.

Example:	At #120 = 27 MOD 4, the value of variable #120 will be 3.

**Logical product, and -** i# = #j AND #k

The code of operation is **AND**.

As a result of operation, the logical product of variables #j and #k will be entered in every bit of the 32 bits in variable #i. Wherever 1 is found at each of the identical bit position of two numbers, 1 will be found there in the result, otherwise 0.

***Functions***

**Square root:** #i = SQRT #j

The code of operation is **SQRT**.

As a result of operation, variable #i will assume the square root of variable #j. The value of #j may not be a negative number.

**Sine:** #i = SIN #j
The code of operation is **SIN**.
As a result of operation, variable #i will assume the sine of variable #j. The value of #j always refers to degrees.

**Cosine:** #i = COS #j
The code of operation is **COS**.
As a result of operation, variable #i will assume the cosine of variable #j. The value of #j always refers to degrees.

**Tangent:** #i = TAN #j
The code of operation is **TAN**.
As a result of operation, variable #i will assume the tangent of variable #j. The value of #j always refers to degrees. The value of #j may not be (2n+1)*90°, where n=0, ±1, ±2, ...

**Arc sine:** #i = ASIN #j
The code of the function is **ASIN**.
As a result of operation, variable #i will assume the arc sine of variable #j in degrees. The condition $-1 \leq \#j \leq 1$ must be true. The result, i.e., the value of #i, lies between +90° and -90°.

**Arc cosine:** #i = ACOS #j
The code of the function is **ACOS**.
As a result of operation, variable #i will assume the arc cosine of variable #j in degrees. The condition $-1 \leq \#j \leq 1$ must be true. The result, i.e. the value of #i, lies between 0° and 180°.

**Arc tangent -** #i = ATAN #j
The code of the function is **ATAN**.
As a result of operation, variable #i will assume the arc tangent of variable #j in degrees. The result, i.e. the value of #i, lies between +90° and -90°.

**Exponent with base e:** #i = EXP #j
The code of the function is **EXP**.
As a result of the operation, variable #i will assume the #j-th power of the natural number (e).

**Logarithm natural:** #i = LN #j
The code of the function is **LN**.
As a result of operation, variable #i will assume the logarithm natural of number #j. The value of #j may not be 0 or a negative number.

**Absolute value:** #i = ABS #j
The code of the function is **ABS**.
As a result of operation, variable #i will assume the absolute value of variable #j.

**Conversion from binary into binary-coded decimal:** #i = BCD #j
The code of the function is **BCD**.
As a result of operation, variable #i will assume the BCD value of variable #j. The value range of variable #j is 0 to 99999999.

**Conversion from binary-coded decimal into binary:** #i = BIN #j

The code of the function is **BIN**.

As a result of the operation, variable #i will assume the binary value of variable #j. The value range of variable #j is 0 to 99999999.

**Discard fractions less than 1:** #i = FIX #j

The code of the function is **FIX**.

This operation will discard the fraction of variable #j, and that value will be put in variable #i.

For example,

```
#130 = FIX 4.8 = 4
#131 = FIX –6.7 = –6
```

**Add 1 for fractions less than 1:** #i = FUP #j

The code of the function is **FUP**

This operation will discard the fraction of variable #j, and will add 1 to #j in absolute value.

For example,

```
#130 = FUP 12.1 = 13
#131 = FUP –7.3 = –8
```

### *Complex Arithmetic Operations - Sequence of Execution*

The above-mentioned arithmetic operations and functions can be combined. The sequence of executing the operations, or the precedence rule is function - multiplicative operations - additive operations.

For example,

```
#110 = #111 + #112 * COS #113
                     ───1────
              ───── 2 ───────  Sequence of operations
       ───────── 3 ──────────
```

### *Modifying the Sequence of execution*

The sequence of executing the operations can be modified by the use of brackets **[** and **]**. Brackets can be nested in 5 levels. The control will return error message *3064 BAD MACRO STATEMENT* if a depth over 5 levels is found in the program.

An example of brackets nested in 3 levels:

```
#120 = COS [ [ [#121 - #122] * #123 + #125] * #126]
               ────── 1 ───
               ────────  2 ─────────
              ───────────  3  ─────────────
            ────────────────  4  ──────────────────
       ──────────────────────── 5 ─────────────────
```

The numbers refer to the sequence of executing the operations. Clearly, the above-mentioned rule of precedence is applicable to the sequence of executing the operations at a given level of brackets.

### 22.13.3 Logical Operations

The programming language uses the following logical operations:

| | |
|---|---|
| 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 |
| Less than or equal to | #i **LE** #j |

The variables on both sides of a logical operation can be substituted by formula as well. The above conditional expressions can be used in divergence or iteration instructions IF or WHILE.

☞ *Note*: Since the above conditional expressions are followed by additions and subtractions, the possible errors must be taken into account in respect of the accuracy of decision.

### 22.13.4 Unconditional Divergence: GOTOn

As a result of instruction **GOTO**n, the execution of the program will be resumed unconditionally at the block of the same program with sequence number n. Sequence number n can be substituted for by a variable or a formula. The number of the block, to which the jump is made by instruction GOTO must be put at the beginning of the block. Unless the selected block number is found, error message *3070 NOT EXISTING BLOCK NO. P* will be returned.

### 22.13.5 Conditional Divergence: IF[<conditional expression>] GOTOn

If [<conditional expression>], put mandatorily between square brackets, is satisfied, the execution of the program will be resumed at the block of the same program with sequence number n.
If [<conditional expression>], is not satisfied, the execution of the program will be resumed at the next block.
Error message *3091ERRONEOUS OPERATION WITH #* is returned unless IF is followed by a conditional expression. If the conditional expression includes a syntactic error, error message *3064 BAD MACRO STATEMENT* will be returned.

### 22.13.6 Conditional Instruction: IF[<conditional expression>] - THEN

If [<conditional expression>], is satisfied, the instruction behind THEN will be executed.
If [<conditional expression>], is not satisfied, the execution of the program will be resumed at the next block.
The word THEN can be omitted, the series of instructions

IF[<conditional expression>] instruction

will be equally executed.

### 22.13.7 Iteration: WHILE[<conditional expression>] Dom ... ENDm

As long as [<conditional expression>] is satisfied, the blocks following DOm up to block ENDm will be repeatedly executed. In the instruction, the control will check wether the condition has been fulfilled; if so, the program detail between DOm and ENDm will be executed; then, as a result of instruction ENDm, the program will return to check the post-WHILE condition again. Unless [<conditional expression>] is satisfied, the execution of the program will be resumed at the block behind ENDm.
If WHILE [<conditional expression>] is omitted, i.e., the cycle is described by instructions DOm ... ENDm, the program detail between DOm and ENDm will be executed for an indefinite (infinite) period of time.
Possible values of m are 1, 2, 3. Error message *3091ERRONEOUS OPERATION WITH #* will be returned if any other value is specified. Error message *3091ERRONEOUS OPERATION WITH #* is returned unless WHILE is followed by a conditional expression. Error message *3064 BAD MACRO STATEMENT* will be returned if the conditional expression includes a syntactic error.

The rules of cycle organization:

– Instruction DOm has to be specified before instruction ENDm.

```
:
END1
:
:                  false (ERROR 72)
:
DO1
```

– Instructions DOm and ENDm must be put in pairs.

```
:
DO1
:
DO1                false
:
END1
:

or

:
DO1
:
END1               false
:
END1
:
```

– A particular identifier number can be used several times.

```
:
DO1
:
END1
:
:                  correct
:
DO1
:
END1
:
```

– Pairs DOm ... ENDm can be nested into one another at three levels.

```
:
DO1
:
DO2
:
DO3
:
:                  correct
:
END3
:
END2
:
END1
:
```

– Pairs DOm ... ENDm may not be overlapped.

```
:
DO1
:
DO2
:
:                  false
:
END1
:
END2
```

– A divergence can be made outside from a cycle.

```
:
DO1
:
GOTO150
:
:                  correct
:
END1
:
N150
:
```

– No entry is permissible into a cycle from outside.

```
    :
    GOTO150
    :
    DO1
    :
    :                  false
    :
    N150
    :
    END1
    :

  or

    :
    DO1
    :
    N150
    :
    :                  false
    :
    END1
    :
    GOTO150
    :
```

– A subprogram or a macro can be called from the inside of a cycle. The cycles inside the subprogram or the user macro can again be nested up to three levels.

```
    :
    DO1
    :
    M98...             correct
    :
    G65...             correct
    :
    G66...             correct
    :
    G67...             correct
    :
    END1
    :
```

### 22.13.8 Data Output Commands

The control will recognize the following data output commands:

| | |
|---|---|
| **POPEN** | periphery open |
| **BPRNT** | binary data print (output) |
| **DPRNT** | decimal data print (output) |
| **PCLOS** | periphery close |

Those data output commands can be used for outputting characters and values of variables. The output may be accomplished to the memory of the control or to an external data storage device (through a serial channel).

### Opening a peripheral - POPENn

Before issuing a data output command, the appropriate peripheral has to be opened, through which the data output is to be performed. The appropriate peripheral is selected by number n.

n = 1   RS–232C interface of serial channel
n = 31  memory of control

A % character is also output to the peripheral simultaneously with the opening of the peripheral, i.e., each data output begins with a % character.

### Binary data output - BPRNT[...]

```
BPRNT[ a #b [c] ... ]
       │  │  └──────────────  number of digits below the decimal point
       │  └─────────────────  variable
       └────────────────────  character
```

The command will send the characters in ISO or ASCII code (depending on the parameter setting); the variables will be output in binary form.

– The characters are output in ISO or ASCII code. The characters to be output are
  alphabetic characters (A, B, ..., Z)
  numerical characters (1, 2, ..., 0)
  special characters (*, /, +, –)
  The control will output the ISO code of a space character (A0h) instead of *.

– The values of variables will be output by the control in 4 bytes (i.e. in 32 bits), beginning with the most significant byte. The number of variables must be followed by the number of digits behind the decimal point in square brackets [ ]. Now the control will convert the floating-point value of the variable into a fixed-point one, in which the number of significant decimal digits are equal to the value put in [ ] square brackets. The possible values of c are 1, 2, ..., 8.
  If, e.g., #120 = 258.647673 and [3] ——— 258648=0003F258h will be output.

– A vacant variable will be output with binary code 00000000h.

– At the end of a data output, the control will automatically output a **L**ine**F**eed character.

For example,

```
BPRNT [ C*/ X#110 [3] Y#120 [3] M#112 [0] ]
   #110=318.49362  ──────   318494=0004DC1Eh
   #120=0.723415  ──────    723=000002D3h
   #112=23.9     ──────     24=00000018h
```

Characters to be output are

```
 7 6 5 4 3 2 1 0
───────────────

 1 1 0 0 0 0 1 1  --- C
 1 0 1 0 0 0 0 0  --- Space
 1 0 1 0 1 1 1 1  --- /
 1 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 0 1 0  --- Line Feed
```

## Decimal data output - DPRNT[...]

```
DPRNT[ a #b [ c d ] ... ]
       │   │   │ │
       │   │   │ └──────────── Number of digits behind the decimal point
       │   │   └────────────── Number of digits before the decimal point
       │   └────────────────── Variable
       └────────────────────── Character
```

All characters and digits will be output in ISO or ASCII code, depending on the parameter setting.

– For the rules of character outputs, see instruction **BPRNT**.

– For the output of variable values, the numbers of decimal integers and fractions must be specified, in which the variable is to be out put. The digits have to be specified in square brackets [ ]. The condition 0 < c + d < 9 must be fulfilled for the specification of digits. The procedure of outputting the digits begins with the most significant digit. In outputting the digits, the negative sign (-) and the decimal point (.) will also be output with the respective ISO codes. If parameter PRNT=0, a space code will be output in the position of the + sign and the leading zeros; each zero is output with code 0 after the decimal point (if any). If parameter PRNT=1, the + sign and the leading zeros will not be output; if the decimal point is defined, the zeros behind it will be output. Otherwise, neither the decimal point nor any of zeros will be output.

– If d=0, the decimal point will be output; if c only is specified, even the decimal point will not be output either.

– A vacant variable will be output with code 0.

– At the end of data outputting, the control will automatically output a line feed character (LF).

Example:

```
DPRNT [ X#130 [53] Y#500 [53] T#10 [2] ]
      #130=35.897421  ─────   35.897
      #500=-150.8   ───────   -150.8
      #10=214.8   ──────────  15
```

Output of data with PRNT=0:

```
 7 6 5 4 3 2 1 0
 _______________
 1 1 0 1 1 0 0 0  --- X
 1 0 1 0 0 0 0 0  --- Space
 1 0 1 0 0 0 0 0  --- Space
 1 0 1 0 0 0 0 0  --- Space
 1 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 (.)
 1 0 1 1 1 0 0 0  --- 8
 0 0 1 1 1 0 0 1  --- 9
 1 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 (–)
 1 0 1 0 0 0 0 0  --- Space
 1 0 1 0 0 0 0 0  --- Space
 1 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(.)
 1 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
 1 1 0 1 0 1 0 0  --- T
 1 0 1 0 0 0 0 0  --- Space
 1 0 1 1 0 0 0 1  --- 1
 0 0 1 1 0 1 0 1  --- 5
 0 0 0 0 1 0 1 0  --- Line Feed (LF)
```

Data output at PRNT=1:

```
 7 6 5 4 3 2 1 0
 _______________
 1 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 (.)
 1 0 1 1 1 0 0 0  --- 8
 0 0 1 1 1 0 0 1  --- 9
 1 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 (–)
 1 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 (.)
 1 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
 1 1 0 1 0 1 0 0  --- T
 1 0 1 1 0 0 0 1  --- 1
 0 0 1 1 0 1 0 1  --- 5
 0 0 0 0 1 0 1 0  --- Line Feed (LF)
```

### Closing a peripheral - PCLOSn

The peripheral opened with command POPEN has to be closed with command PCLOS. Command PCLOS has to be followed by the specification of the number of peripheral to be closed. At the time of closing, a % character is also sent to the peripheral, i.e., each data output is terminated by a % character.

☞ *Notes*:
– The sequence of data output commands is a fixed one. First the appropriate peripheral has to be opened with command POPEN, followed by the process of data outputting (with command BPRNT or DPRINT); finally, the open peripheral has to be closed with instruction PCLOS.
– The opening and closing of a peripheral can be specified in any point of the program. For example, it can be opened and closed at the beginning and end of the program, respectively, data can be output in any part of the program in between.
– A command M30 or M2 executed during the process of data output will interrupt the data transfer. To avoid this, waiting is to be performed during data transfer before the execution of command M30.
– The parameters (baud rate, number of stop bits etc.) of the peripheral have to be set correctly. They can be selected in group SERIAL of the field of parameters.

## 22.14 NC and Macro Instructions

NC and macro blocks can be differentiated in the programming language. The blocks written in terms of conventional codes G, M etc. are regarded as NC blocks even when the values of the addresses assume variables or formulae as well as numerical values.
The following blocks are regarded as macro instructions:
– The block containing a definition, substitution instruction (#i=#j)
– a block containing a conditional divergence or iteration instruction (IF, WHILE)
– blocks containing control commands (GOTO, DO, END)
– blocks containing macro calls (G65, G66, G66.1, G67, or codes G, or M that initiate macro calls).

## 22.15 Execution of NC and Macro Instructions in Time

The macro blocks can be executed by the control parallel to NC blocks or in consecutive order. Parameter SBSTM determines the execution of NC and macro blocks. If the parameter:
=0: NC and macro blocks are executed in the order written in the program,
=1: Macro statements are executed in the course of NC block execution

Example:

**SBSTM**=0

```
%O1000
...
N10 #100=50
N20 #101=100
N30 G1 X#100 Y#101
N40 #100=60 (definition after N30)
N50 #101=120 (definition after N30)
N60 G1 X#100 Y#101
```

Definition commands in blocks N40 and N50 are executed after the movement of block N30.

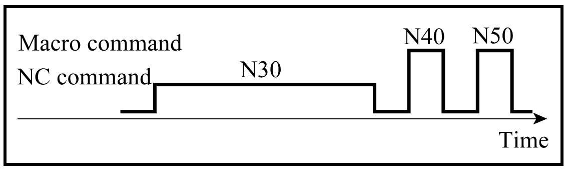


Fig. **22.15-1**

☞ *Conclusions:*

*– Program execution is slower,*

*– if execution of block N30 is interrupted and afterwards the machining is restarted the machining can be simply continued since variables of block N30 are not overwritten by block N40, N50.*

**SBSTM**=1

```
%O1000
...
N10 #100=50
N20 #101=100
N30 G1 X#100 Y#101
N40 #100=60 (definition during N30)
N50 #101=120 (definition during N30)
N60 G1 X#100 Y#101
```

Definition commands in blocks N40 and N50 are executed during movement in block N30.

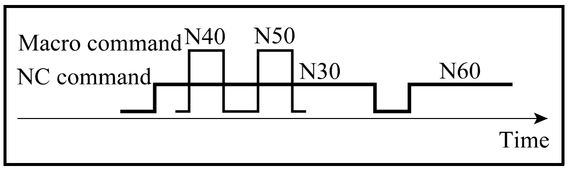


Fig. **22.15-2**

☞ *Conclusions:*

*– Program execution is faster,*

*– if execution of block N30 is interrupted and afterwards the machining is restarted the machining can not be continued, only if block search is started for block N30 since variables of block N30 are already overwritten by the blocks N40, N50.*

## 22.16 Displaying Macros and Subprograms in Automatic Mode

The blocks of macros and subprograms will be displayed by the control in automatic mode. If parameter *MD8* is set to 0, the blocks of subprograms and macros numbered 8000 to 8999 will not be listed when they are executed. With parameter *MD8* set to 1, their blocks will also be listed. If parameter *MD9* is set to 0, the blocks of subprograms and macros numbered 9000 to 9999 will not be listed when they are executed. With parameter *MD9* set to 1, their blocks will also be listed.

## 22.17 Using the STOP Button While a Macro Instruction is Being Executed

Pressing the STOP button, i.e., suspension of the program execution will be effective always on completion of the macro instruction being executed.

## Notes

## Index in Alphabetical Order: