NCT® 99M

NCT® 2000M

Controls for Milling Machines and Machining Centers

Programmer's Manual

Manufactured by **NCT Automation kft**.
H1148 Budapest Fogarasi út 7
: Address: H1631 Bp. pf.: 26
F Phone: (+36 1) 467 63 00
F Fax:(+36 1) 363 6605
E-mail: nct@nct.hu
Home Page: www.nct.hu

Contents

July 2, 2002

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	angular coordinates, length 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
H, D	number of length and radius compensation cell	1 - 99
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 Sub-program

The part programs may be divided into two main groups -

main programs, and

subprograms.

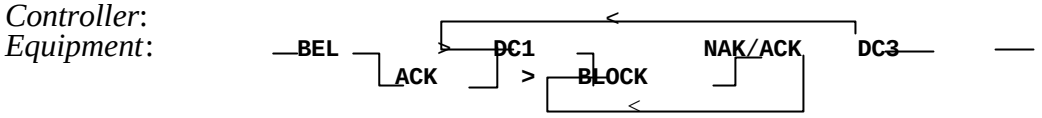
The procedure of machining a part is described in the main program. If, in the course of machining repeated patterns have to be machined at different places, it is not necessary to write those program-sections over and over again in the main program, instead, a sub-program has to be organized, which can be called from any place (even from another sub-program). The user can

return from the sub-program 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 ┗F) 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 Y__
X__ Y__
```

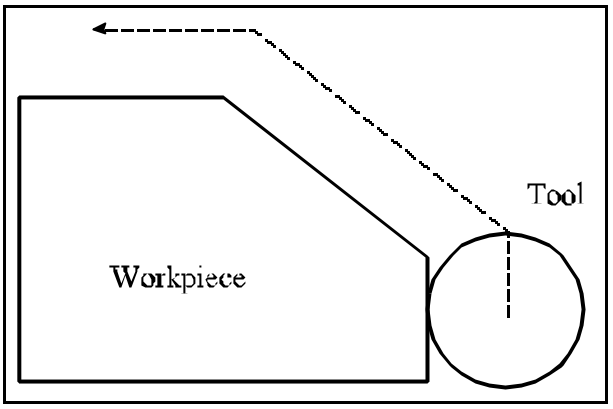
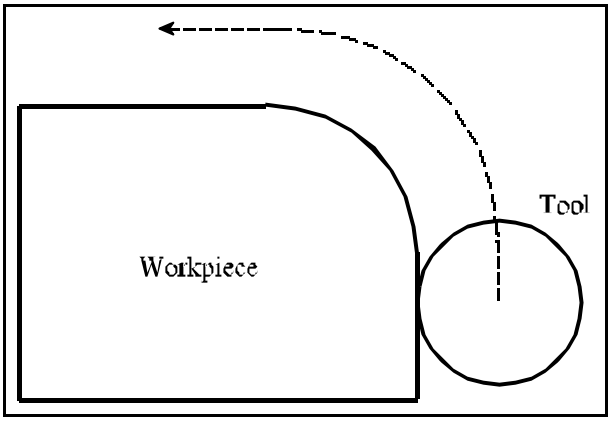


Fig. 1.2-1

Tool movement along an arc:

program:

```
G03 X__ Y__ R__
```



Although, in general, the table with the workpiece and not the tool moves, this description will refer to the motion of the tool against the workpiece.

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 F150 means 150 mm/minute.

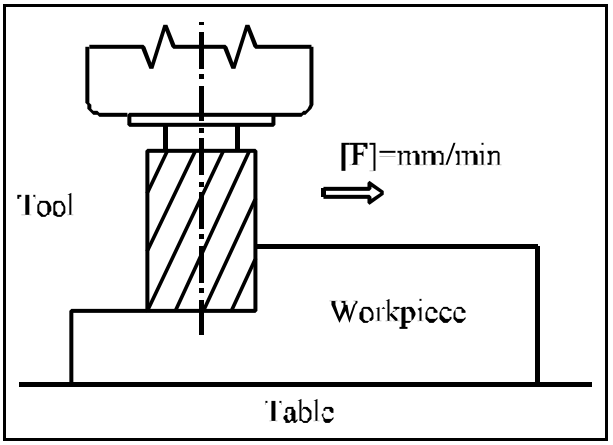


Fig. 1.2-3

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., X340 means a point of coordinate 340 mm in the coordinate system of the workpiece.

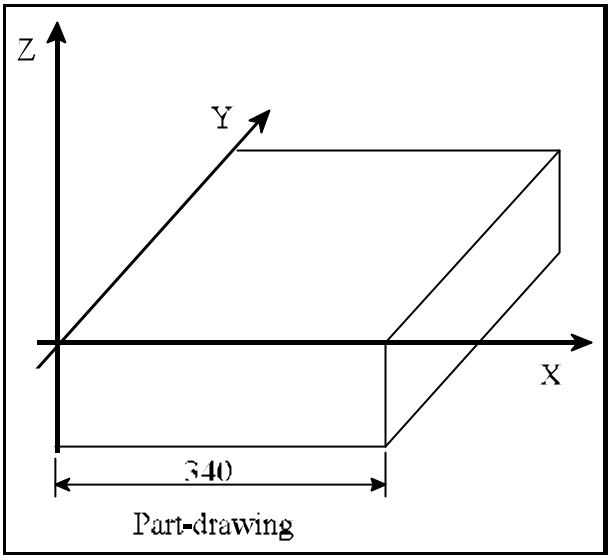


Fig. 1.2-4

The coordinate system specified in the control system and 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 points of the two coordinate systems have to be set at the same position. This can be achieved, e.g., by moving the tool center to a point of known position of the part and setting the coordinate system of the control to that value.

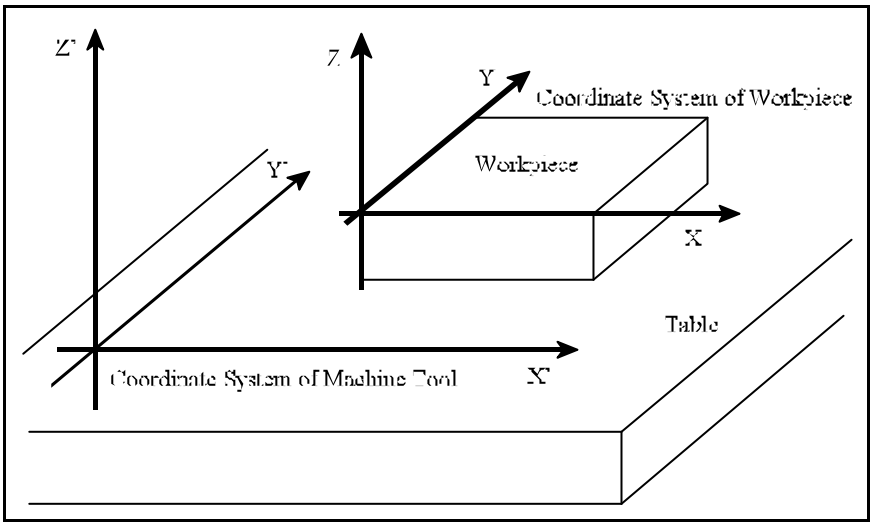


Fig. 1.2-5

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.

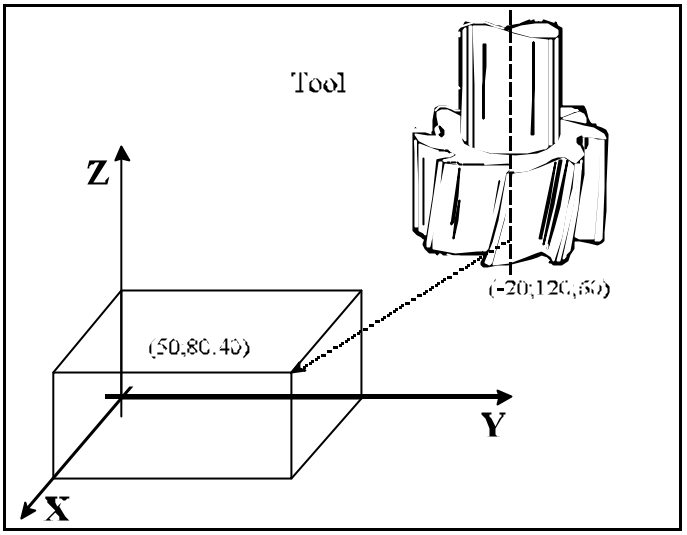


Fig. 1.2-6

The code of absolute data specification is G90.
The block

```
G90 X50 Y80 Z40
```

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

Incremental Coordinate Specification

In the case of an incremental data specification, the control system will interpret the coordinate data in such a way that the tool will travel a distance measured from its instantaneous position.
The code of incremental data specification is G91. Code G91 refers to all coordinate values.
The block

```
G91 X70 Y-40 Z-20
```

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

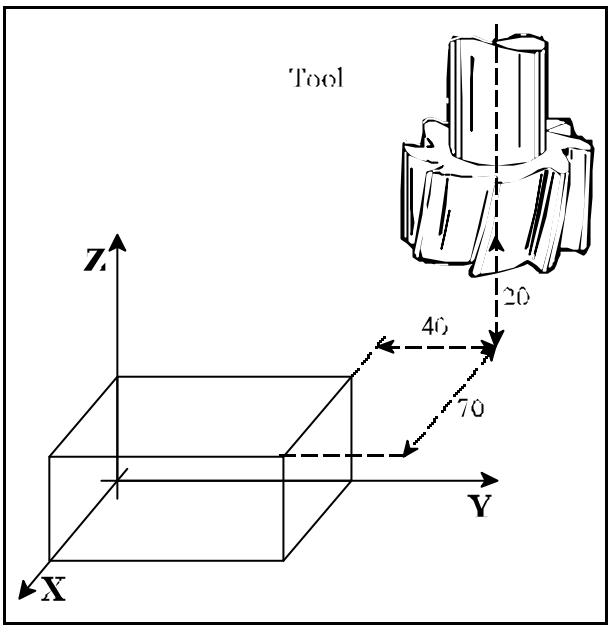


Fig. 1.2-7

An incremental data may be defined to be referred to a single coordinate as well. Standing behind the address of the coordinate, character I refers to the incremental specification of the given coordinate value.
In block

```
G90 XI-70 Y80 Z40
```

the data of X is interpreted as an incremental value, whereas data Y and Z are - for code G90 - interpreted as absolute coordinates.

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 Y30 F180
N16 X30
N17 Y100
```

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

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. Instruction T25 in the program means that tool No. 25 has to be changed. The tool change can be carried out manually or automatically, depending on the design of the machine.

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 address H. E.g., instruction H1 refers to length data No.1. Henceforth the control will move the tip of the tool to the specified point. That procedure is referred to as setting “tool length compensation" mode.

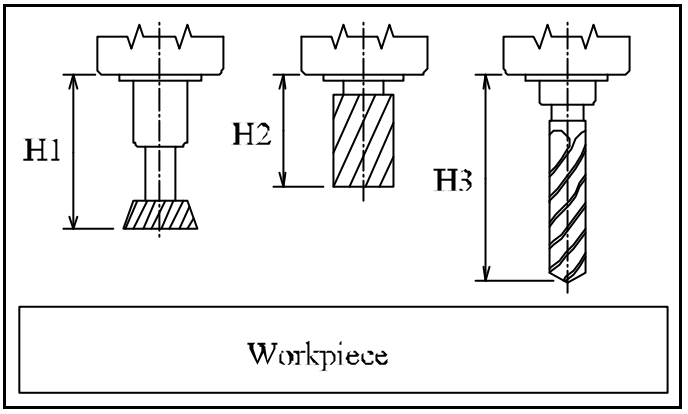


Fig. 1.2-8

Cutter Radius Compensation

Machining a workpiece has to be done with tools of different radii. 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 center (taking into consideration the tool radii). The values of radius compensations have to be set in control system. Hereinafter reference can be made to cutter compensations at address D in the program.

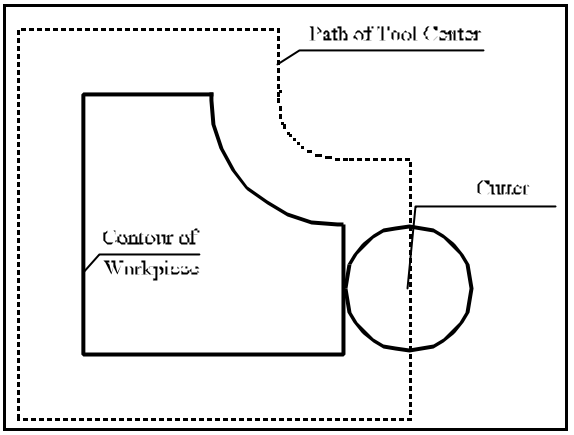


Fig. 1.2-9

Wear Compensation

The tools are exposed to wear in the course of machining. Allowance can be made for such dimensional changes (in length and radius as well) with wear compensations. The tool wear can be set in the control system. A geometry value, i.e., the initial length and radius of the tool, and a wear one belong to each compensation group (referred to at address H or D). When the compensation is set, the control will compensate the movement with the sum of the two values.

2 Controlled Axes

Number of Axes (in basic configuration)	3 axes
In expanded configuration	5 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 in a milling control system: X, Y and Z.
The names of additional (expansion) axes depend on their respective types.
Possible names of expansion axes performing linear motions are: U,V and W. When they 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.

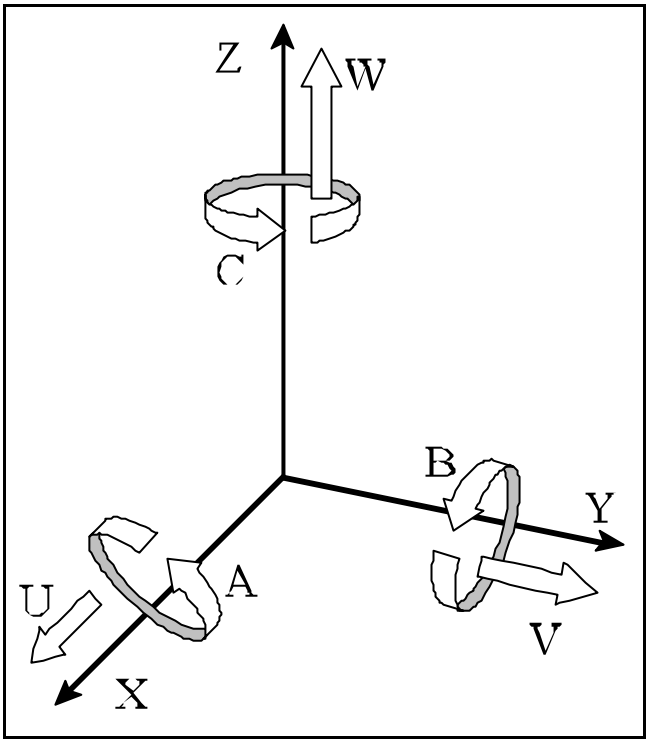


Fig. 2.1-1

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 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 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

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	22
G01*		linear interpolation	22
G02		circular, helical interpolation, clockwise (CW)	24
G03		circular, helical interpolation, counter-clockwise (CCW)	24
G04	00	dwell	52
G05.1		multi-buffer mode on	
G07.1		Cylindrical interpolation	35
G09		exact stop (in the given block)	49
G10		data setting (programmed)	59, 79
G11		programmed data setting cancel	
G12.1	26	Polar coordinate interpolation on	31
G13.1		Polarc coordinate interpolation off	31
G15*	17	polar coordinate command cancel	38
G16		polar coordinate command	38
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	40
G21		metric input	40
G22*	04	programable stroke check function on	158
G23		programable stroke check function off	158
G25*	8	spindle speed fluctuation detection off	67
G26		spindle speed fluctuation detection on	67
G28	0	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	155
G33	01	thread cutting	29
G37	00	Automatic tool-length measurement	156
G38		cutter compensation vector hold	100

G code	Group	Function	Page
G39		cutter compensation corner arc	100
G40*	07	cutter radius/3 dimensional tool compensation cancel	85
G41		cutter radius compensation left/3 dimensional tool compensation	85, 88
G42		cutter radius compensation right	85, 88
G43*	08	tool length compensation +	80
G44*		tool length compensation –	80
G45	00	tool offset increase	81
G46		tool offset decrease	81
G47		tool offset double increase	81
G48		tool offset double decrease	81
G49*	08	tool length compensation cancel	80
G50*	11	scaling cancel	116
G51		scaling	116
G50.1*	18	programable mirror image cancel	117
G51.1		programable mirror image	117
G52	00	local coordinate system setting	60
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	49
G62		automatic corner override mode	50
G63		override inhibit	50
G64*		continuous cutting	50
G65		simple macro call	161
G66		macro modal call (A) in every motion command	162
G66.1		macro modal (B) call from each block	163
G67		macro modal call (A/B) cancel	162
G68	16	coordinate system rotation	115
G69*		coordinate system rotation cancel	115
G73	09	High Speed Peck Drilling Cycle	138
G74		counter tapping cycle	139
G76		fine boring cycle	140

G code	Group	Function	Page
G80*		canned cycle cancel	141
G81		drilling, spot boring cycle,	141
G82		drilling, counter boring cycle	142
G83		peck drilling cycle	143
G84		tapping cycle	144
G84.2		rigid tap cycle	145
G84.3		rigid counter tap cycle	145
G85		boring cycle	148
G86		Boring Cycle Tool Retraction with Rapid Traverse	149
G87		Boring Cycle/Back Boring Cycle	150
G88		Boring Cycle (Manual Operation on the Bottom Point)	152
G89		Boring Cycle (Dwell on the Bottom Point, Retraction with Feed)	153
G90*	03	absolute command	38
G91*		incremental command	38
G92	00	work coordinates change/maximum spindle speed setting	59
G94*	05	feed per minute	46
G95*		feed per revolution	46
G96	13	constant surface speed control	65
G97*		constant surface speed control cancel	65
G98*	10	canned cycle initial level return	133
G99		canned cycle R point level return	133

L *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)
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.

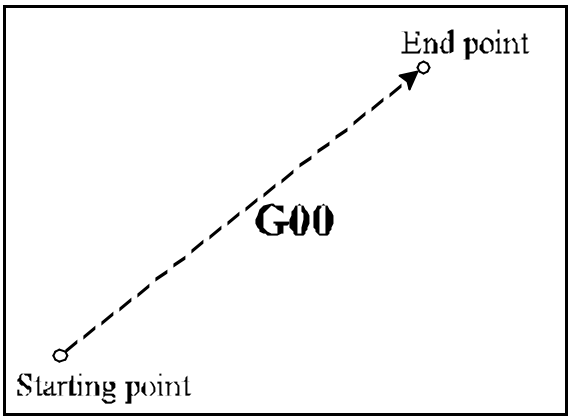


Fig. 4.1-1

In executing the G00 instruction, the control system performs acceleration and declaration in starting and ending the movements, respectively. 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* of the parameter field.

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

Feed along the axis Y is $F_y = \frac{y}{L} F$

.....

Feed along the axis U is $F_u = \frac{u}{L} F$

.....

Feed along the axis C is $F_c = \frac{c}{L} F$

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

$$L = \sqrt{x^2 + y^2 + \ldots + u^2 + \ldots + c^2}$$

```
G01 X100 Y80 F150
```

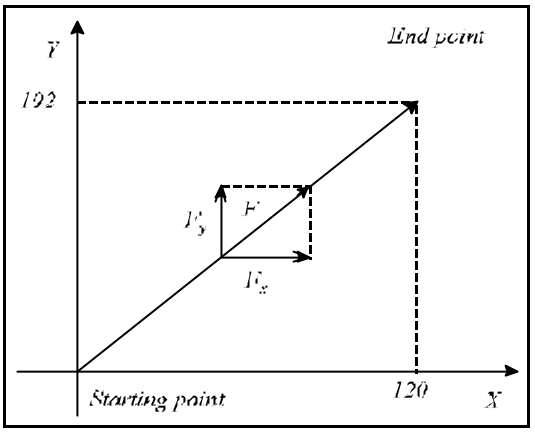


Fig. 4.2-1

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

```
G01 B270 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 B45 F120
```

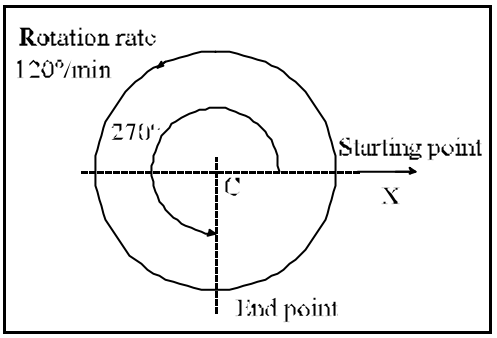


Fig. 4.2-2

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 B: $F_b = \frac{45}{\sqrt{100^2 + 45^2}} 120 = 49.2$ °/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)

$$G17 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} X_p\ Y_p \left\{\begin{matrix} R \\ I\ J \end{matrix}\right\} F$$

$$G18 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} X_p\ Z_p \left\{\begin{matrix} R \\ I\ K \end{matrix}\right\} F$$

$$G19 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} Y_p\ Z_p \left\{\begin{matrix} R \\ J\ K \end{matrix}\right\} F$$

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).

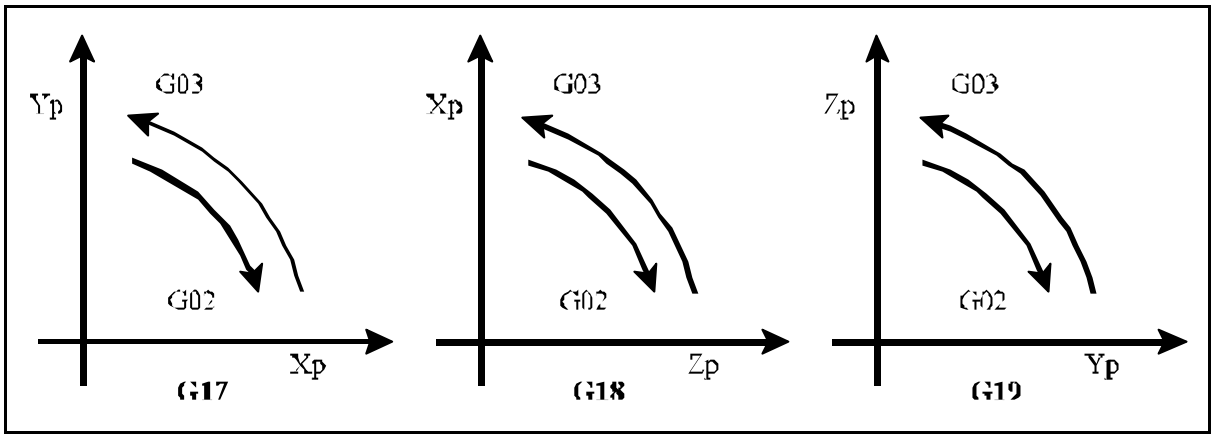


Fig. 4.3-1

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:

Arc section 1: `G02 X50 Y40 R40`
Arc section 2: `G02 X50 Y40 R-40`
Arc section 3: `G03 X50 Y40 R40`
Arc section 4: `G03 X50 Y40 R-40`

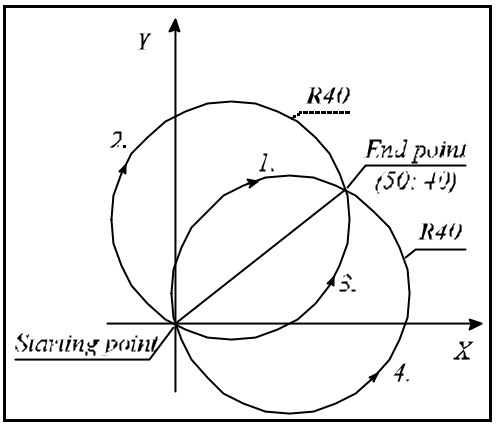


Fig. 4.3-2

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. For example:

With G17: `G03 X10 Y70 I-50 J-20`
With G18: `G03 X70 Z10 I-20 K-50`
With G19: `G03 Y10 Z70 J-50 K-20`

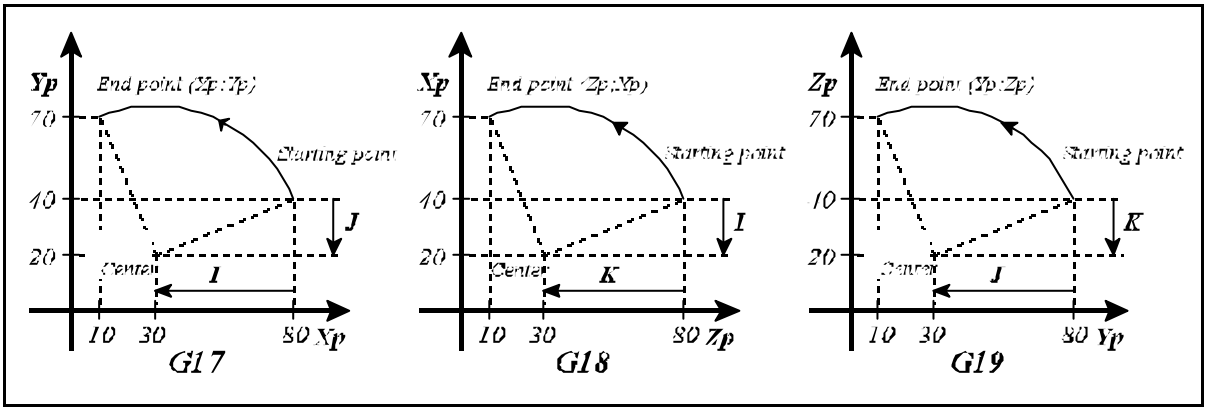


Fig. 4.3-3

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.

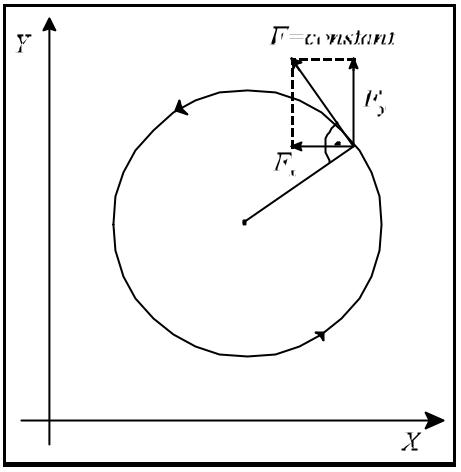


Fig. 4.3-4

L *Notes*:

– I0, J0, K0 may be omitted, e.g.

```
G03 X0 Y100 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.

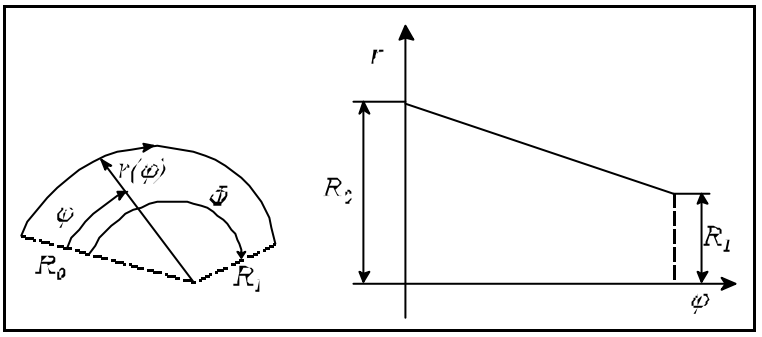


Fig. 4.3-5

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, J, K.

```
G17 G90 G0 X50 Y0
G3 X-20 I-50
```

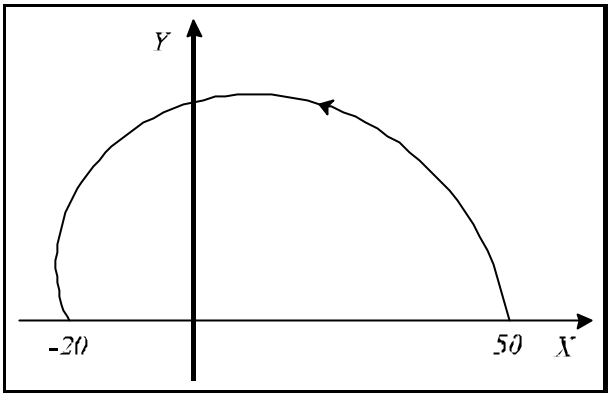


Fig. 4.3-6

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.

```
G17 G0 G90 X0 Y0
G2 X40 Y30 R10
```

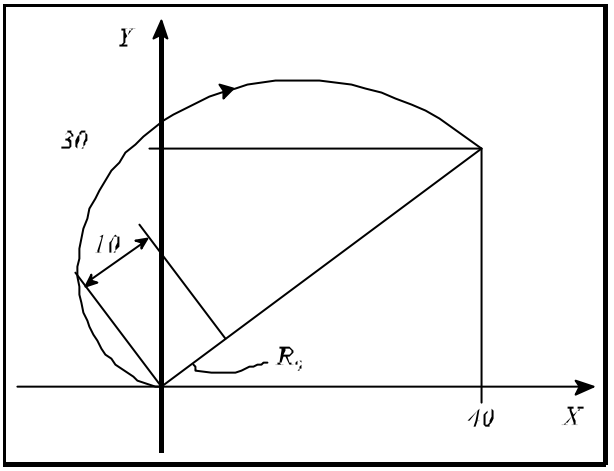


Fig. 4.3-7

4.4 Helical Interpolation (G02, G03)

$$G17 \left\{\frac{G02}{G03}\right\} X_p Y_p q \left\{\frac{R}{I\ J}\right\} F$$

$$G18 \left\{\frac{G02}{G03}\right\} X_p Z_p q \left\{\frac{R}{I\ K}\right\} F$$

$$G19 \left\{\frac{G02}{G03}\right\} Y_p Z_p q \left\{\frac{R}{J\ K}\right\} F$$

The series of instructions will define a helical interpolation.
It is distinguished from circular interpolation that a third axis (q), which is not an axis composing the circular plane. The control performs a simple movement along axis q.

The feed specified at address F is effective along the circle path. Feed component Fq along axis q is obtained from the relationship

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

where

L_q: displacement along axis q,
L_{arc}: length of circular arc,
F: programmed feed,
F_q: feed along axis q.

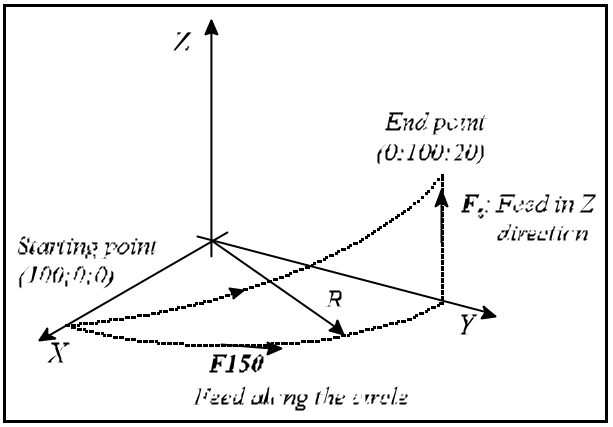


Fig. 4.4-1

For example:

```
G17 G03 X0 Y100 Z20 R100 F150
```

The series of instructions

$$G17 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} X_p\ Y_p\ q\ r\ s \left\{\begin{matrix} R \\ I\ J \end{matrix}\right\} F$$

$$G18 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} X_p\ Z_p\ q\ r\ s \left\{\begin{matrix} R \\ I\ K \end{matrix}\right\} F$$

$$G19 \left\{\begin{matrix} G02 \\ G03 \end{matrix}\right\} Y_p\ Z_p\ q\ r\ s \left\{\begin{matrix} R \\ J\ K \end{matrix}\right\} F$$

define a multi-dimensional spatial helical interpolation in which q, r, s are optional axes not involved in the circle interpolation.

For example, series of instructions

```
G17 G3 X0 Y-100 Z50 V20 I-100
```

will move the tool along the superficies of an oblique cylinder if V is an axis parallel to Y.

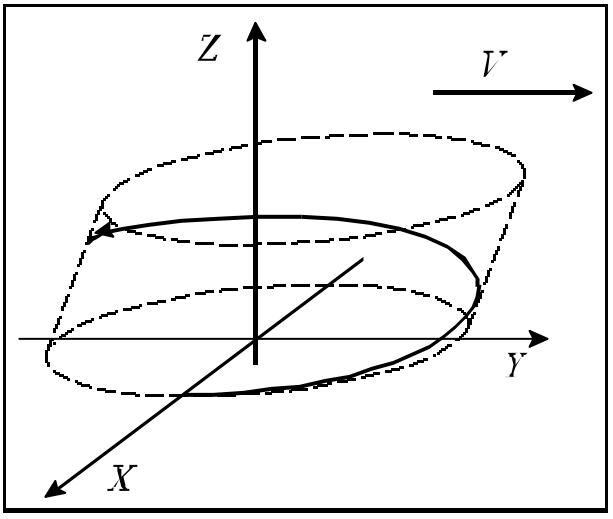


Fig. 4.4-2

Notes:

– Whenever parameter *HELICALF* in the field of parameters is set to 1, the control will implement the programmed feed along the spatial path.

– In the case of the circle specified in the selected plane having a varying radius, the interpolation will be carried out along the superficies of the specified cone.

– The specified tool-radius compensation is implemented invariably in the plane of the circle.

4.5 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 "<45°, i.e. Z>X, the programmed lead will be taken into account along axis Z,

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

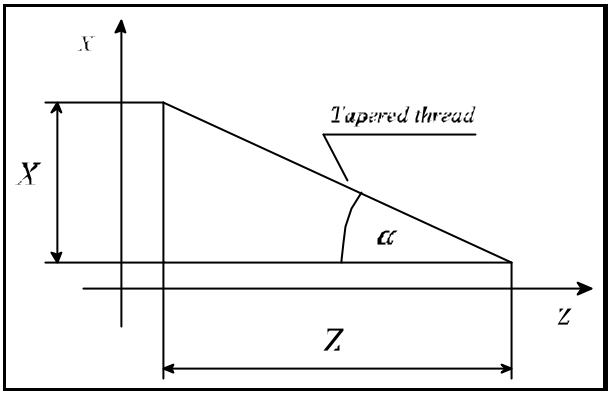


Fig. 4.5-1

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.

– 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.

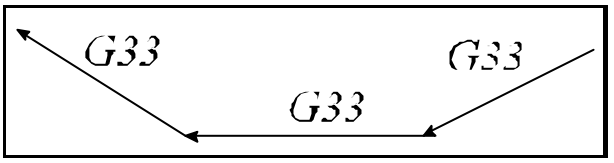


Fig. 4.5-2

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:

```
N50 G90 G0 X0 Y0 S100 M4
N55 Z2
N60 G33 Z-100 F2
N65 M19
N70 G0 X5
N75 Z2 M0
N80 X0 M4
N85 G4 P2
N90 G33 Z-100 F2
...
```

Explanation:

N50, N55 - Moving the tool over the center of hole, starting the spindle in counter-clockwise rotation,
N60 - First thread-cutting cycle, (lead 2mm),
N65 - Oriented spindle stop (the spindle is stopped in a fixed position),
N70 - Tool retraction along axis X,
N75 - Tool retraction to the top of hole, programmed stop, the operator adjust the tool to the next thread-cutting cycle,
N80 - Return to the center of hole, re-start of spindle,
N85 - Waiting for the speed to be assumed by the spindle,
N90 - Second thread-cutting cycle.

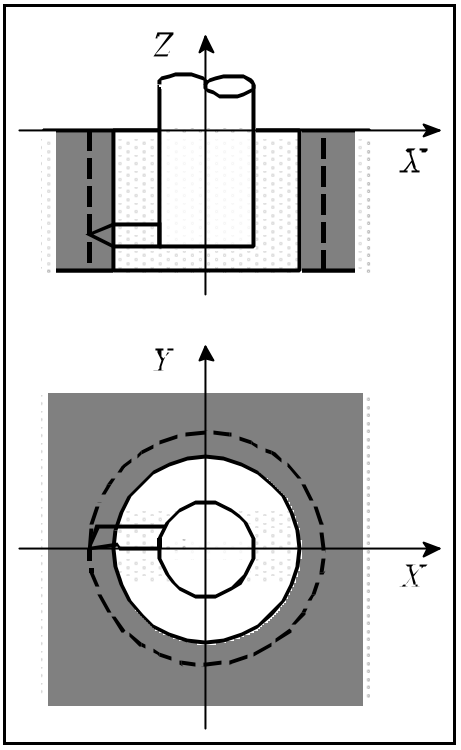


Fig. 4.5-3

L *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).

4.6 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.

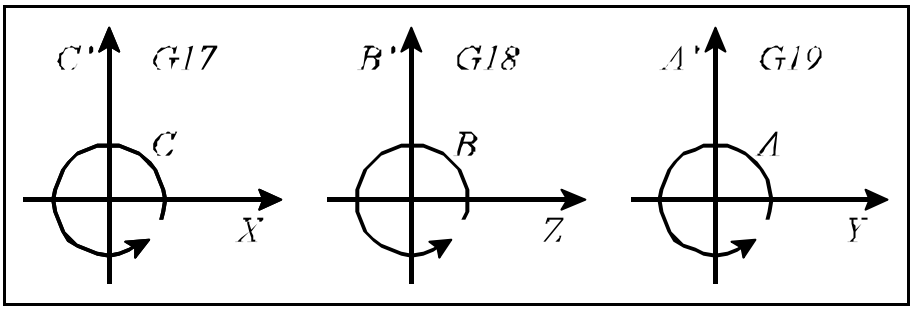


Fig. **4.6-1**

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 must be chosen so that it coincides with the rotation axis of the circular axis.

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 polat 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. 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.) x move belongs to the programmed feed within a time unit. Different angular moves (n_1, n_2, n_3, n_4) belong to) x 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.

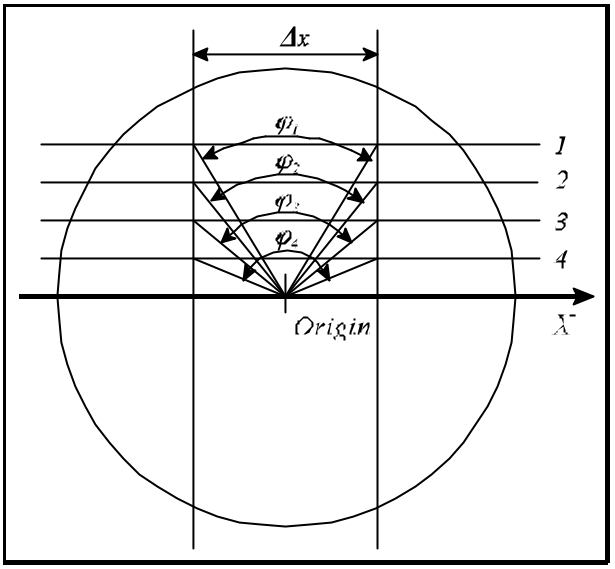


Fig. 4.6-2

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.

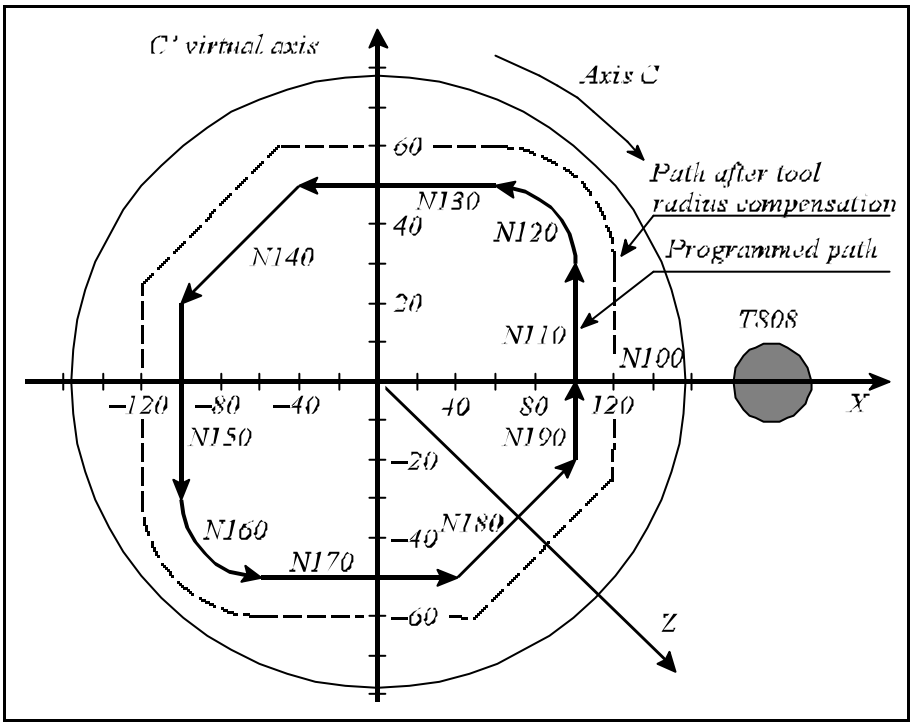


Fig. 4.6-3

```
%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

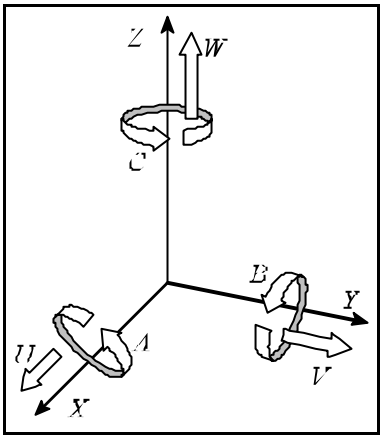


Fig. 4.7-1

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:

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

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

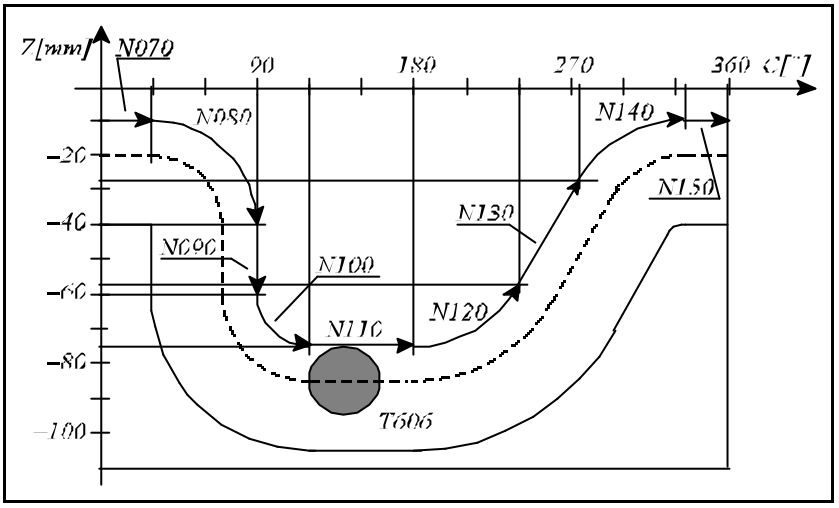


Fig. 4.7-2

```
%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 *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 X20 Y50
G91 G01 X-40 Y30
```

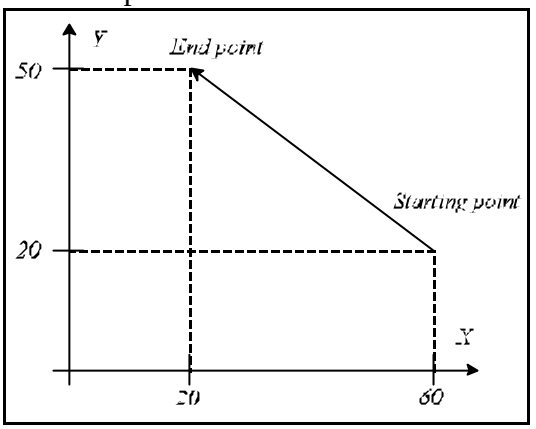


Fig. 5.1-1

Operator I will be effective under the conditions of an absolute data specification. 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 XI-40 YI30
G01 X20 YI30
G01 XI-40 Y50
```

5.2 Polar Coordinates Data Command (G15, G16)

Alternatively, the coordinates of the end point can be entered with polar coordinate data specification, i.e., with the specification of angle and radius.

G16: Polar coordinate data command

G15: Polar coordinate data command cancel

The control is in G15 state after power on. G15 and G16 are modal functions.

The data of polar coordinates are effective in the plane defined by G17, G18, G19. When a data is specified, the addresses of the plane's horizontal and vertical axes are regarded as radius and angle, respectively. For example, in G17 sate, the data written at addresses X(U) and Y(V) are the radius and angle, respectively. CAUTION! In state G18, Z and X are the horizontal and the vertical axes (data of R and angle, respectively).

When an angular data is specified, the positive and the negative directions of the angle are counter-clockwise and clockwise, respectively.

The data of the rest of axes will be assumed to be Cartesian coordinate data. The radius and the angle can be specified both as absolute and as incremental data.

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

Example:

```
G90 G16 G01 X100 Y60 F180
```

Both the radius and the angle are absolute data, the tool moves to the point of 100mm; 60°.

```
G90 G16 G01 X100 YI40 F180
```

The angle is an incremental data. A movement by 40° relative to the previous angular position is moved.
With the radius, specified as an incremental value, the instantaneous position of the axes will be the origin of the polar coordinate system.
A circle can be programmed with polar coordinate data command (G16). The circle can be also specified with the radius and I, J, K as well. In the latter case, however, the control will regard addresses I, J, K invariably as Cartesian data. When the origin of the current coordinate system coincides with the center of a circle or a helix, a multiple turn one can also be programmed with polar coordinate data specification.

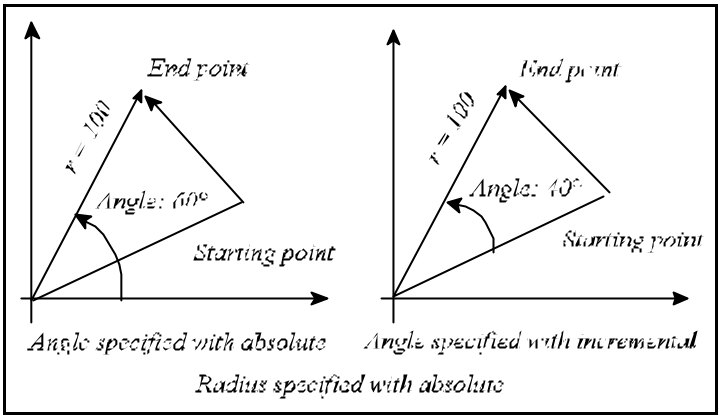


Fig. 5.2-1

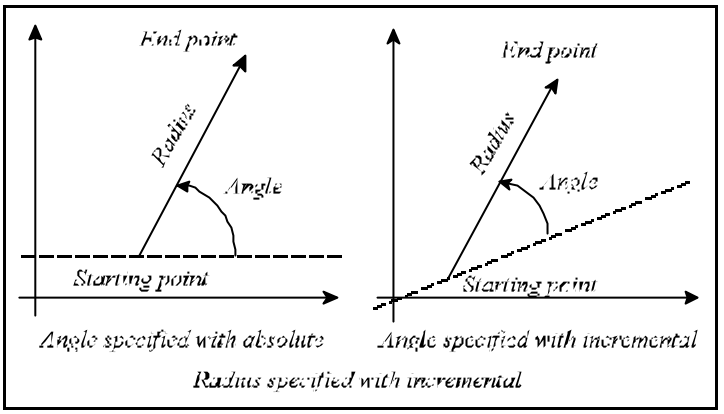


Fig. 5.2-2

Example:

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

A helix of 2¾ turns has been specified in the above block in counter-clockwise direction of rotation.
In programming a multiple-turn circle, bear in mind that a negative or a positive polar angle has to be programmed for direction G2 or G3, respectively.

L *Notes*:

The addresses encountered in the following instructions will not be regarded as polar coordinate specifications even when state G16 is:

– G10 coordinates encountered in setting instruction,
– G52 coordinate offset,
– G92 coordinate setting,
– G53 positioning in machine coordinate system,
– G68 coordinate rotation,
– G51 scaling on,
– G50.1 programmable mirror image.

An example of milling a hexagon:

```
N1 G90 G17 G0 X60 Y0 F120
N2 G16 G1 Y60
```

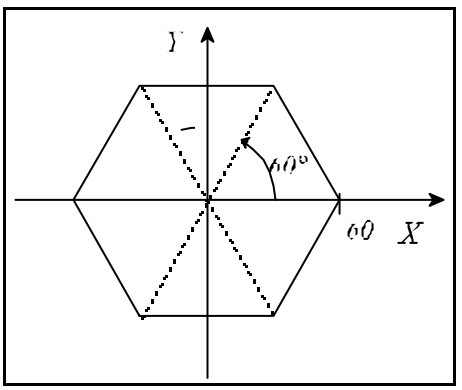


Fig. 5.2-3

```
N3 Y120
N4 Y180
N5 Y240
N6 Y300
N7 Y360
N8 G15 G0 X100
```

5.3 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.*

5.4 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	

5.5 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 ROLLOVEN 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 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 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, 0243 **ROLLOVEN_C**=1 0263 **ROLLAMNT_C** = =360000	Block programmed by incremental data input	Movement affected by block	Position at block end
0249 **RELROUND_C**=0 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 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 travers

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.

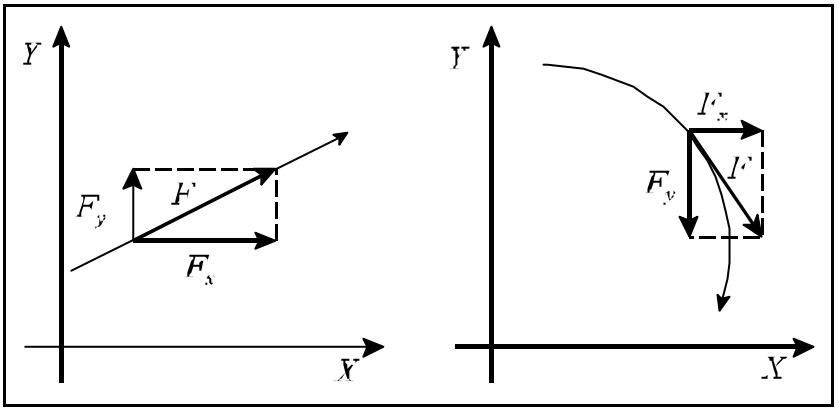


Fig. 6.2-1

F - tangential feed (programmed value)
F_x - feed component in the X direction
F_y - feed component in the Y direction

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

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 Automatic Acceleration/Deceleration

In rapid traverse, the control will automatically perform a linear acceleration and linear deceleration when starting and ending a movement. The extent of acceleration is defined by the machine tool builder, in parameter *ACCn*, depending on the dynamics of the machine.

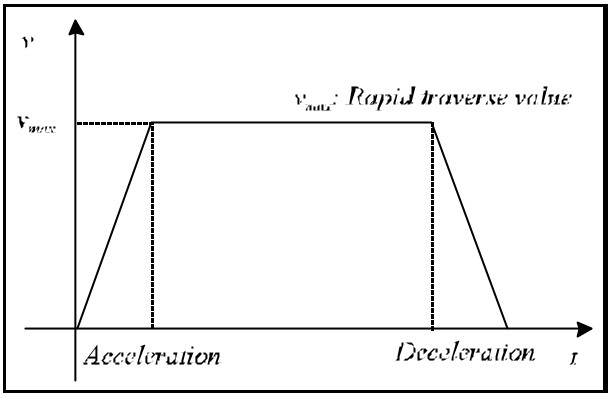


Fig. 6.3-1

In feed motions the tangential (programmed) feed value will be assumed by the control in linear acceleration, inversely, its value will be decreased by linear deceleration. This technique offers the advantage over traditional (exponential) accelerations that the machine will sooner attain the desired speed (assuming a given time constant adopted in both cases). Thus the times of acceleration and deceleration (i.e., the times of actual slide movements) will be reduced.

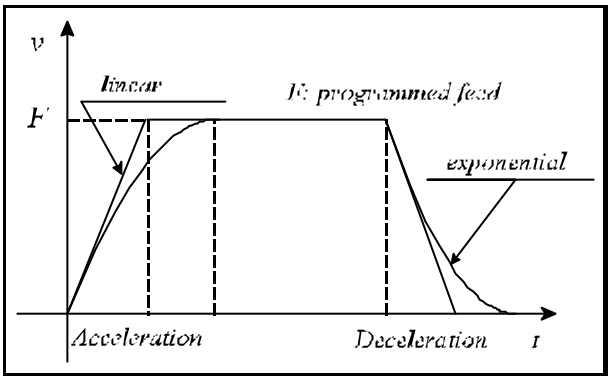


Fig. 6.3-2

Another advantage of linear acceleration over the exponential one is the lower profile distortion (i.e., radius error), compared with exponential acceleration, in a high-speed machining of a circle.

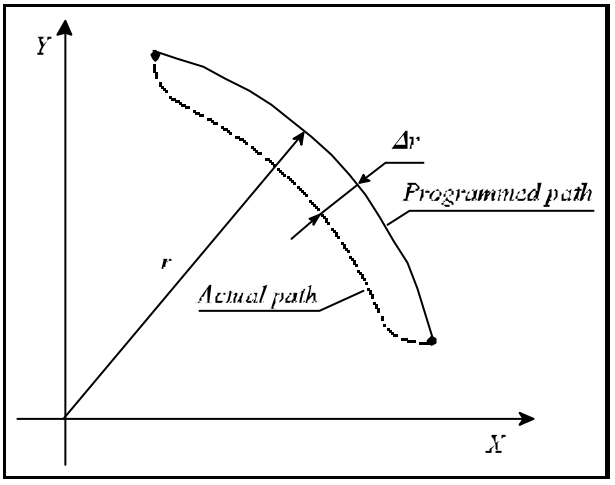


Fig. 6.3-3

The control is monitoring the changes in tangential speeds. This is necessary to attain the commanded speed in a process of continuous acceleration, if necessary, through several blocks. The acceleration to the new feed (higher than the previous one) is commenced by the control invariably in the execution of the particular block, in which the new feed value is specified. That process may, if necessary, cover several blocks. Deceleration to the new feed value (lower than the previous one) will be started by the control in an appropriate preceding block so that the machining will be started with the programmed speed in the particular block, in which the new feed value is specified.

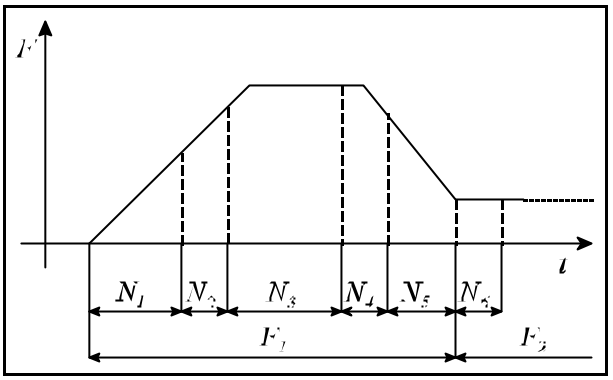


Fig. 6.3-4

When moving manually by using jog keys or handwheel, again linear acceleration/deceleration will be performed. Their values will be defined for each axis by parameters *ACC1* through ACC8.

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.

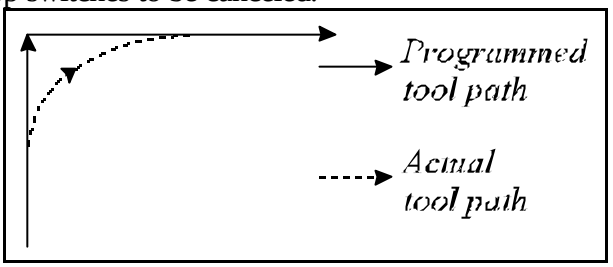


Fig. 6.4-1

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.

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 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 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. The corner override is effective under the following conditions.

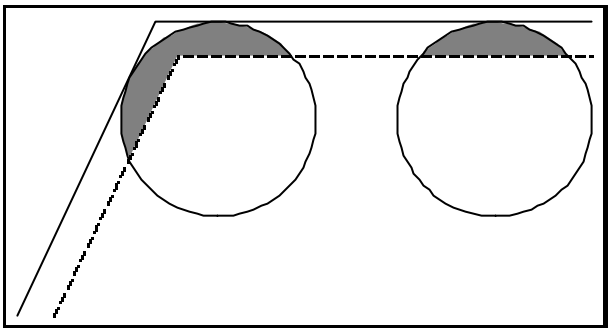


Fig. 6.4.5-1

– When cutter 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 1 can be selected between 1 and 180° by parameter *CORNANGLE*.

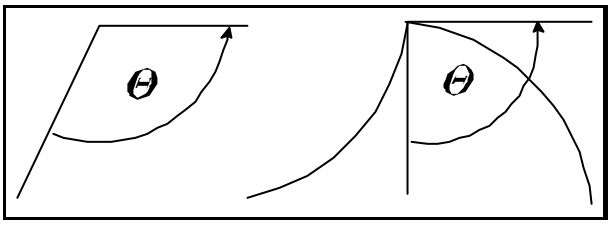


Fig. 6.4.5-2

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.

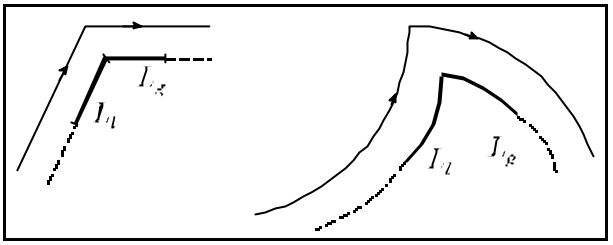


Fig. 6.4.5-3

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 cutter 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

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

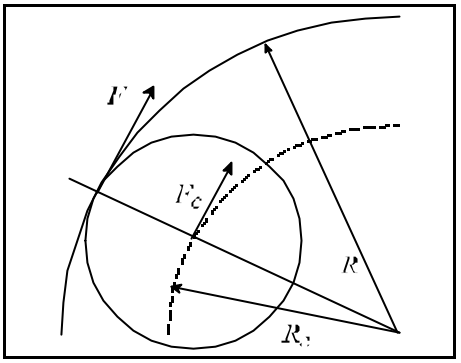


Fig. 6.4.6-1

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.

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.

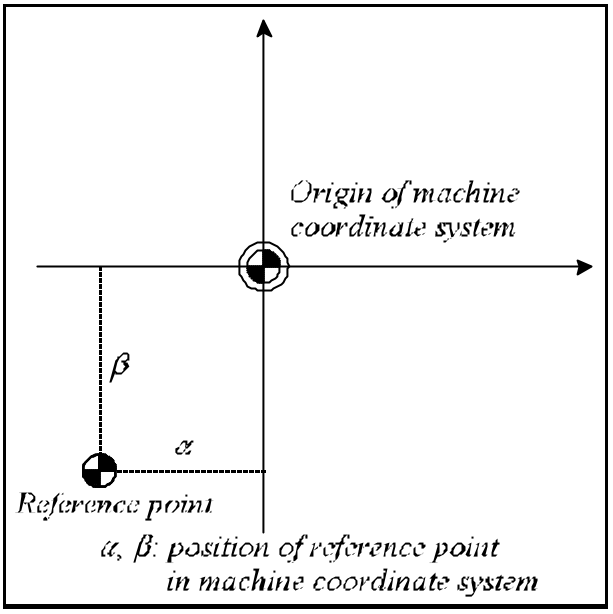


Fig. 8-1

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 cutter 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.

L *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, Y=0
G28 Y200 intermediate point: X=100, Y=200

8.2 Automatic return to reference points 2nd, 3rd, 4th (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 cutter 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 cutter 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 intermediate point.
When the cutter 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 X500 Y200
G29 X700 Y150
...
...
```

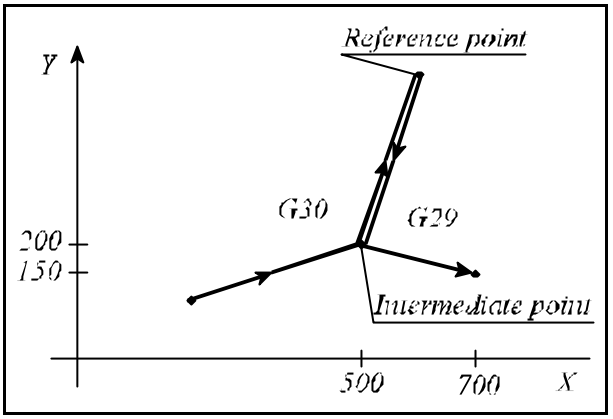


Fig. **8.3-1**

9 Coordinate Systems, Plane Selection

The position, to which the tool is to be moved, is specified with coordinate data in the program. When 3 axes are available (X, Y, Z), the position of the tool is expressed by three coordinate data X____ Y____ Z____ :

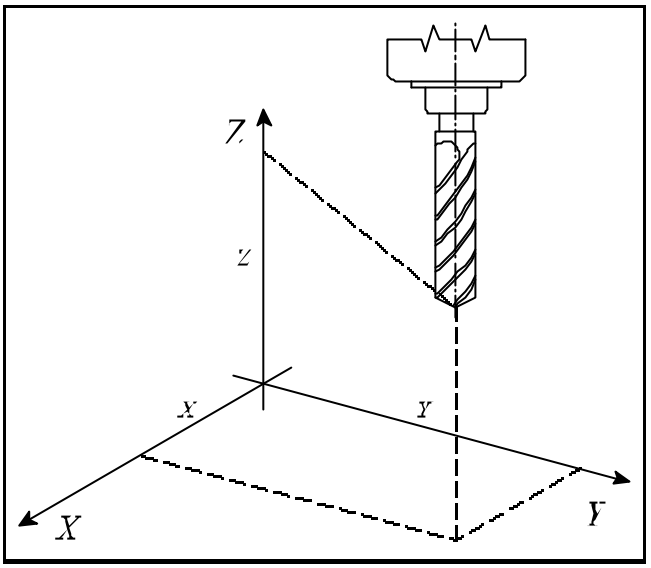


Fig. 9-1

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.

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.

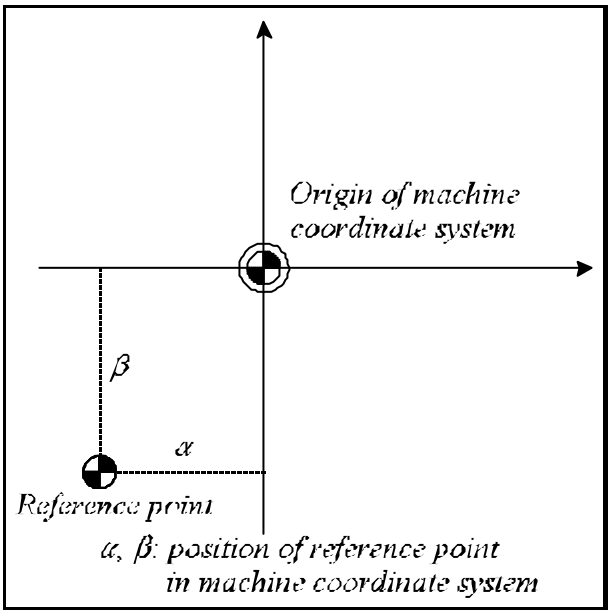


Fig. 9.1-1

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

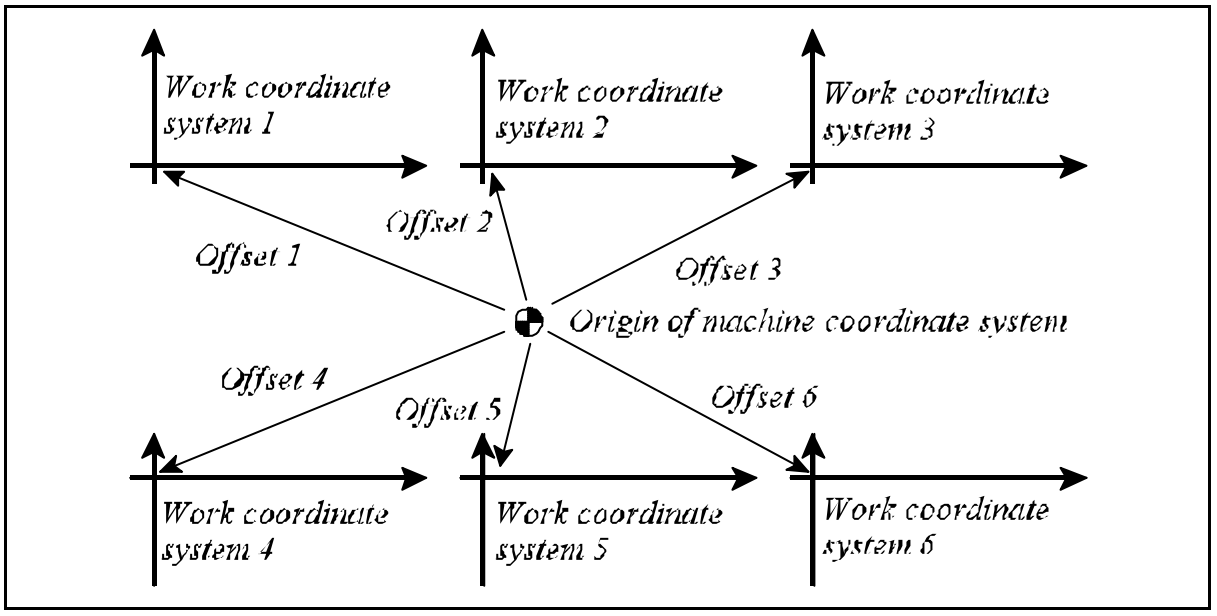


Fig. 9.2.1-1

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.

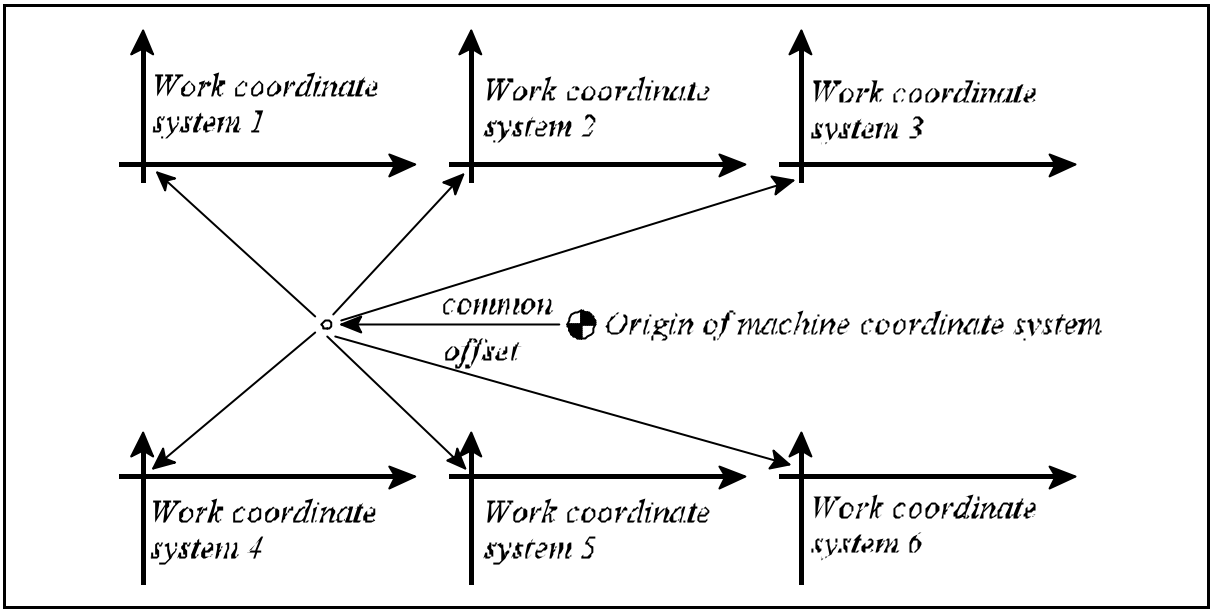


Fig. 9.2.1-2

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 X60 Y40
```

will move the system to point X=60, Y=40 of work coordinate system 3.

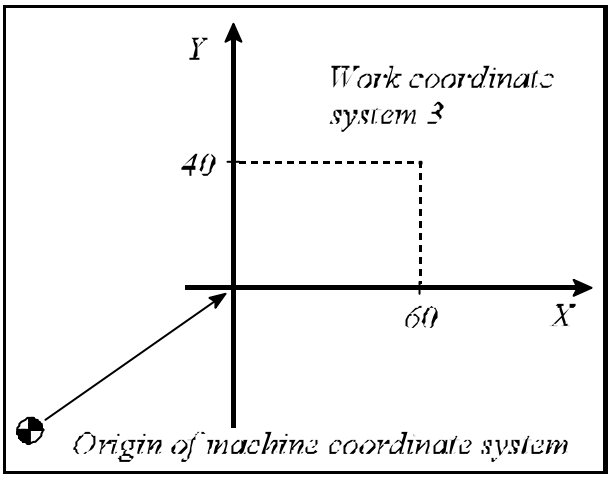


Fig. 9.2.2-1

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=300, Y=800 (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=1300, Y=400 (calculated in the machine coordinate system). The tool position is X'=700, Y'=500 in X', Y's' coordinate system (G54). As a result of instruction G55, the tool position will be interpreted in the X", Y" coordinate system (X"=-300, Y"=900).

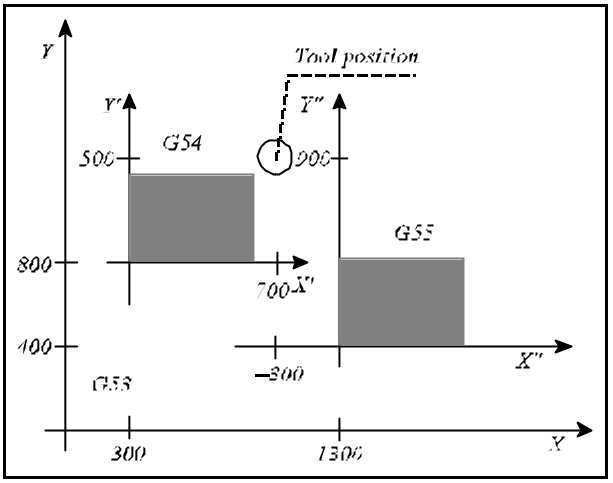


Fig. 9.2.2-2

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 = 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.

If, e.g., the tool is at a point of X=150, Y=100 coordinates, in the actual (current) X, Y work coordinate system, instruction

```
G92 X90 Y60
```

will create a new X', Y' coordinate system, in which the tool will be set to the point of X'=90, Y'=60 coordinates. The axial components of offset vector v' between coordinate systems X, Y and X', Y' are

v'_x=150-90=60,

and

v'_y=100–60=40.

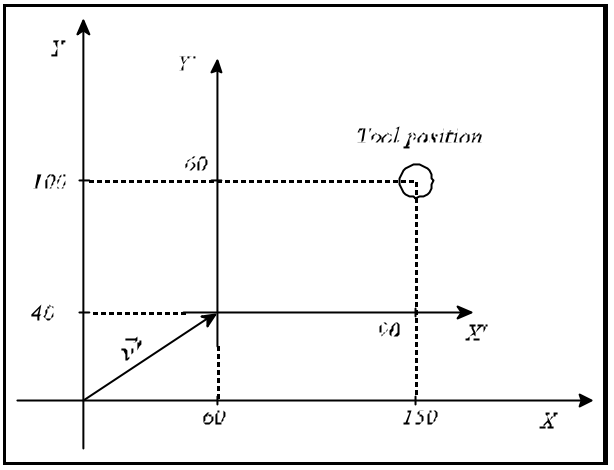


Fig. 9.2.4-1

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.

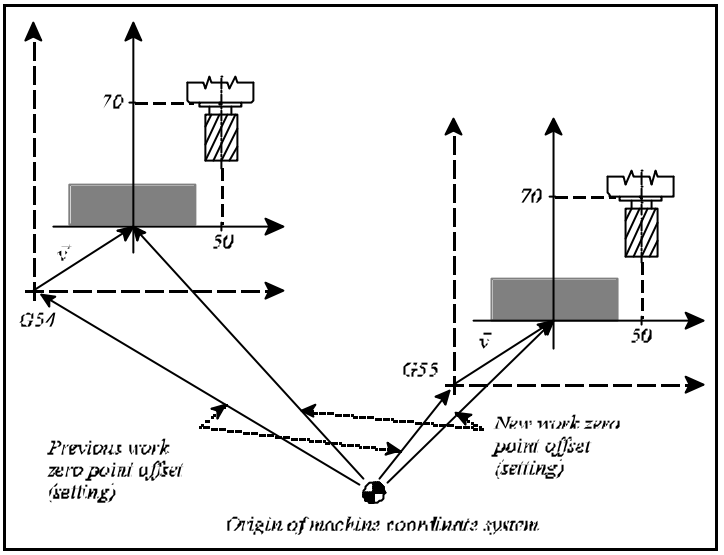


Fig. 9.2.4-2

L *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.

– Instruction G92 offers a convenient way of the cyclic position indication of the indexing rotary table performing several turns. If, e.g., axis B has been turned into the position 360°, the axis can be moved to position 0° without any physical movement by programming G92 B0.

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 part 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=150, Y=100 coordinates in the current X, Y work coordinate system, instruction

```
G90 G52 X60 Y40
```

will create a new local X', Y' coordinate system, in which the coordinates of tool will be X'=90, Y'=60. Instruction G52 is used for defining the axial components of offset vector v' between the X, Y and X', Y' coordinate systems (v'_x=60, v'_y=40).

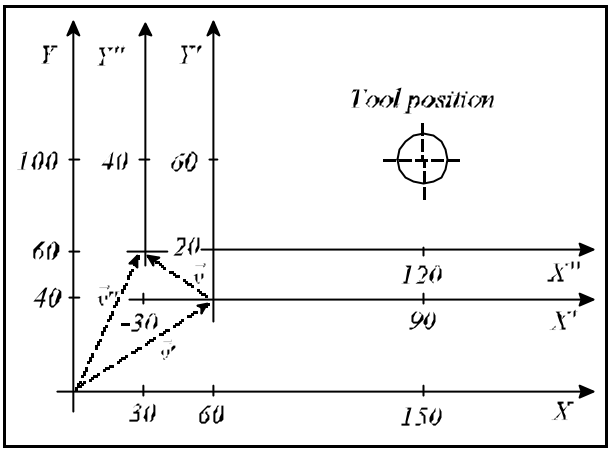


Fig. 9.3-1

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

– With an absolute data specification: instruction (G90) G52 X30 Y60 will move the origin of the X'', Y'' coordinate system to point X=30, Y=60 in the X,Y work coordinate system. The components of vector v'' will be produced by the specification of v''_x=30, v''_y=60.

– With an incremental data specification: instruction G91 G52 X-30 Y20 will move the origin of the X'', Y'' coordinate system to the point of X'=-30, Y'=20 coordinates in the X', Y' coordinate system. The components of vector v will be produced by the specification of v_x=-30, v_y=20. Indicating the location of the new local coordinate system in the X, Y work coordinate system vector v''=v'+v. Its components:
v''_x=60+(-30)=30, v''_y=40+20=60.

The tool position in the X'', Y'' coordinate system will be X''=90, Y''=40.

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.

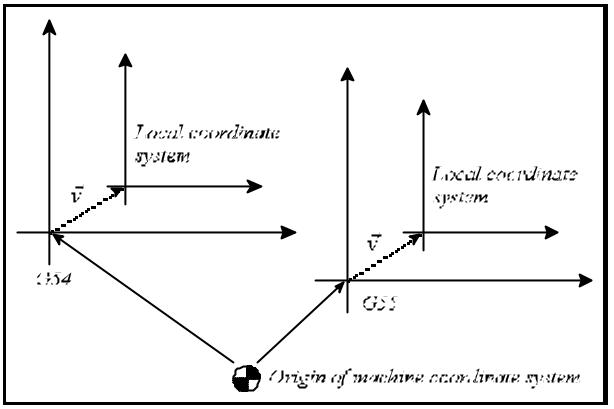


Fig. 9.3-2

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=200, Y=120 coordinates in the X, Y work coordinate system, instruction

```
G52 X60 Y40
```

will shift its position to X'=140, Y'=80 in the X', Y' local coordinate system.

Now instruction

```
G92 X110 Y40
```

will establish the tool position to X''=110, Y''=40 in the new X'', Y'' work coordinate system. Thus the X', Y' local coordinate system will be deleted by command G92 as if command G52 X0 Y0 had been issued.

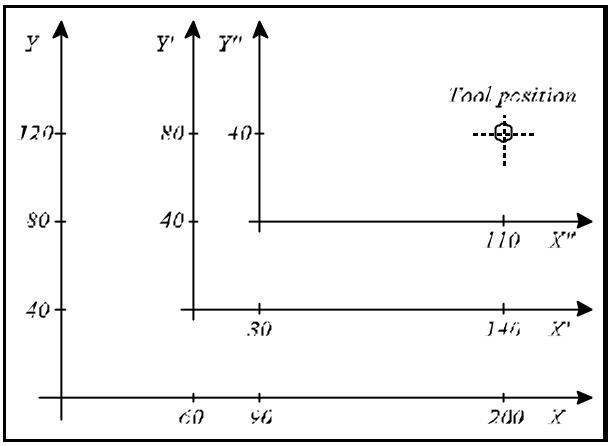


Fig. 9.3-3

L *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,
– specification of polar coordinate data,
– rotation of coordinate system,
–cutter 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

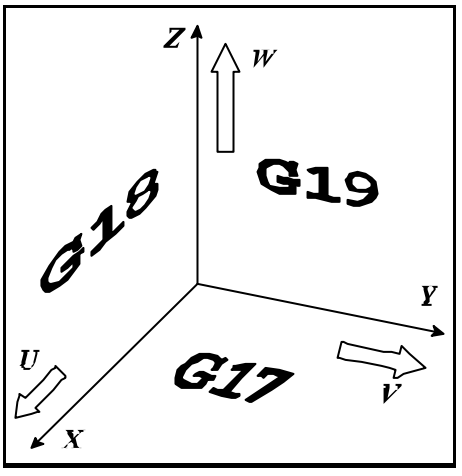


Fig. 9.4-1

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_.

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:

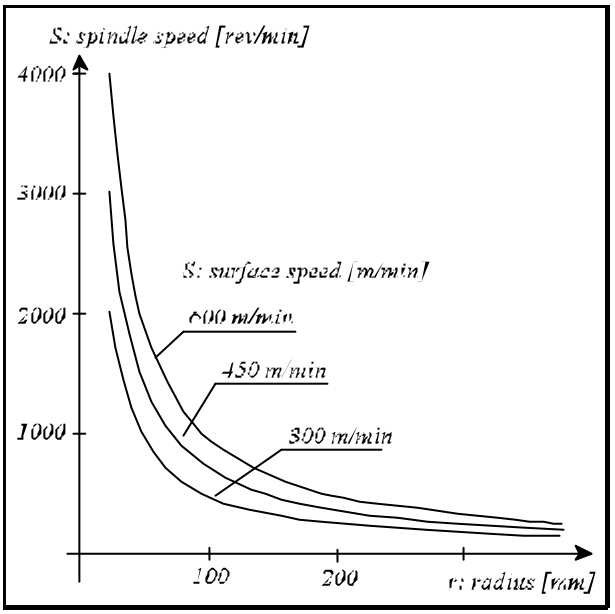


Fig. **10.2-1**

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

10.2.1Constant 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 or until the control is turned off.

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 commands

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 a 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

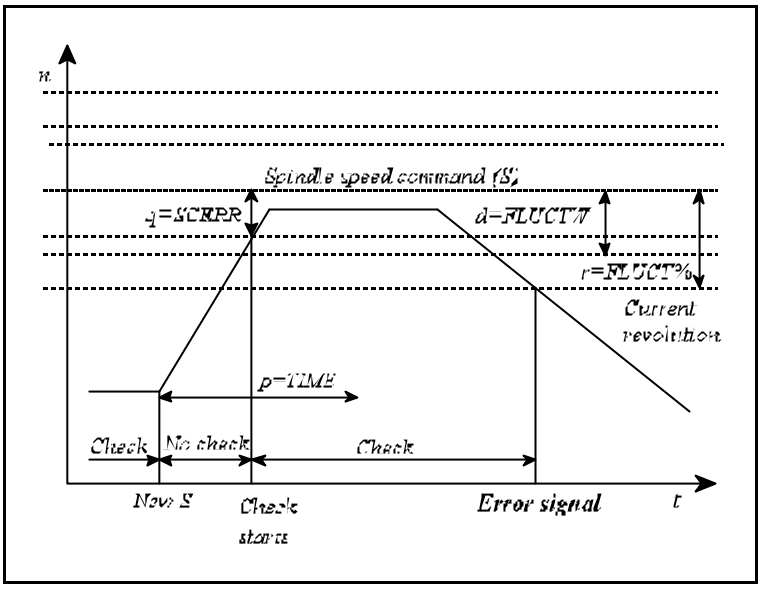


Fig. **10.6-1**

- 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 .

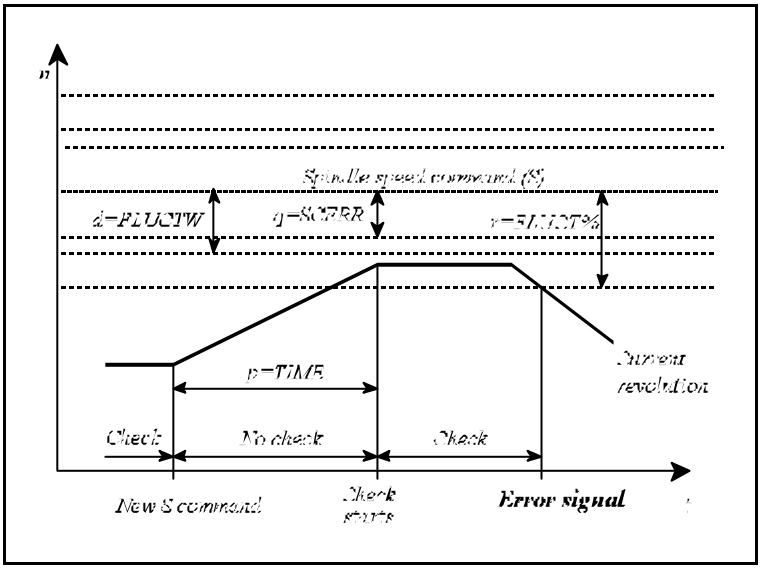


Fig. **10.6-2**

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.

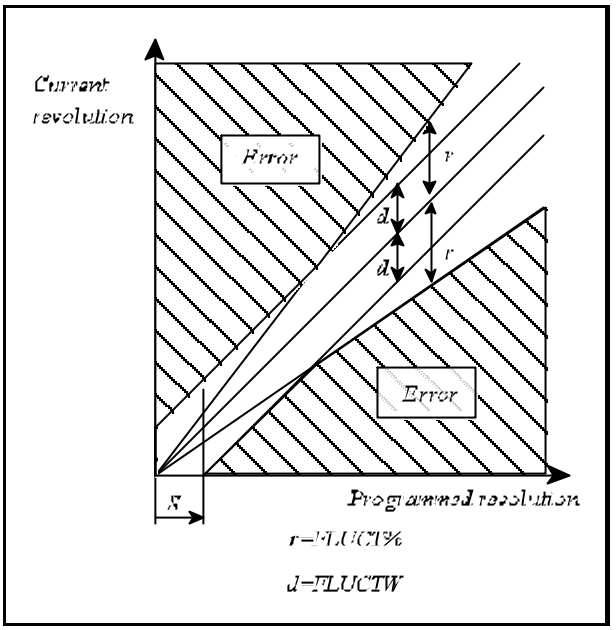


Fig. **10.6-3**

– 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

11.1 Tool Select Command (Code T)

With a number of max. four digits written at address **T**, the NC will give a code to the PLC. When a movement command and a tool number (T) are programmed in a given block, function T will be issued during or after the motion command. The machine tool builder will define the way of execution.

11.2 Program Format for Tool Number Programming

There are basically two different ways of making reference to a tool change in the part program. They depend on the configuration of the machine-tool. The particular technique of calling the tool (applicable in the part program) is defined by the builder of the machine-tool.

Case A

A tool change can be accomplished on the machine manually or by means of a turret type tool changer. Now, with reference made to code T:

– in the case of manual tool change, the number of tool to be used appears on the display; it has to be clamped in the spindle manually. Afterwards, the machining will be resumed upon a start;
– in the case of a turret type tool changer, the new tool will be put to position of use automatically under the action of code T.

Thus a reference to the tool number will evoke an immediate change in the block which T has been specified in.

Case B

A tool change needs some preparations on the machine. Its steps:

– The tool to be employed has to be found in the magazine. Now reference to address T in the part program will bring the appropriate tool in change position. This operation is carried out in the background, simultaneously with the machining.
– The slides (or only one of them) have to be sent to the change positions.
– The tool change is carried out by function M06 in the program. The control will wait for the execution of the tool change until tool T (under preparation) is brought to change position. As a result, the new tool will be placed in the spindle. Henceforth cutting may be resumed.
– The old tool is replaced in the tool magazine. This activity is performed in the background, simultaneously with cutting.
– The search is commenced for the new tool in the magazine.

This procedure is described in the part program as follows.

Part Program	Explanation
.....	
....Tnnnn.....	search for tool Tnnnn
.....	the part program is running, tool search is being performed in the background
...M06 Tmmmm....	tool T*nnnn* is placed in the spindle,
.....	the previous tool is replaced in the tool magazine
.....	the search is commenced for tool T*mmmm*, with cutting performed in the meantime
...M06 Tpppp.....	tool T*mmmm* is placed in the spindle
.....	tool T*nnnn* is replaced in the tool magazine, search for tool T*pppp* begins
.....	meanwhile machining is being performed

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
M06: tool change code
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	M06 (tool exchange)
group 2	M11, ..., M18 (spindle gear range change)
group 3	M03, M04, M05, M19 (spindle management)
group 4	M07, M08, M09 (coolant management)
group 5	M*nnn* (any other function M)
group 6	codes M of spindle indexing
group 7	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 change:	M06
2.	Tool call:	T
3.	Spindle range selection:	M11, ..., M18
4.	Spindle speed: S	
5.	Spindle management:	M03, M04, M05, M19
6.	Coolant:	M07, M08, M09
7.	Other function M:	M*nnn*
8.	Spindle indexing:	with function M
9.	Function A:	A
10.	Function B:	B
11.	Function C:	C
12.	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 n^{th} switch is on,
– that block will be executed when the n^{th} switch is off.

13.3 Main Program and Sub-program

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 sub-program must be terminated with code M99.

13.3.1 Calling the Sub-program

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 sub-program machining will be continued in the main program with the block following the subprogram call.

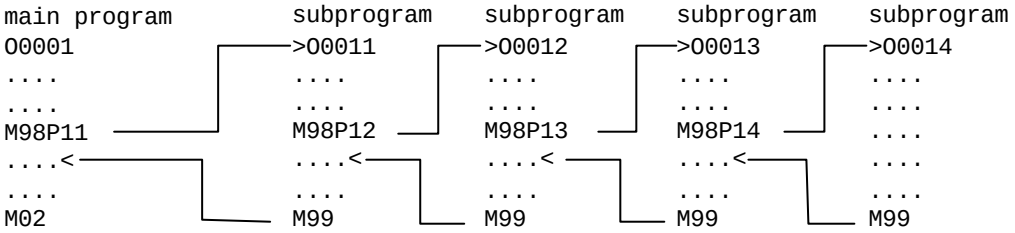
```
main program                        subprogram          comment

O0010                                                   execution of (main-)
......                                                  program O0010
......
M98 P0011           --->            O0011               calling sub-program
                                                        O0011
                                    ......              execution of sub-
                                    ......              program 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 sub-program 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.



L *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 Sub-program

The use of instruction

M99

in a sub-program means the end of that sub-program, 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 sub-program
                                                        O0011
                                    ......              execution of sub-
                                    ......              program O0011
                                    ......
N102 ......           <---          M99                 return to the next
                                                        block of the calling
                                                        program
......                                                  resumption of program
......                                                  O0010
```

The use of instruction

M99 P...

in a sub-program means the end of that sub-program, 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 sub-program
                                                        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 sub-program 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.)

L *Note*:

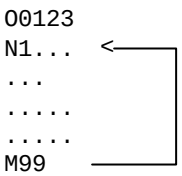
– An error message *3070 NOT EXISTING BLOCK NO. P* is displayed when the return block number (P) is not found in the calling program.

13.3.3 Jump within the Main Program

The use of instruction

M99

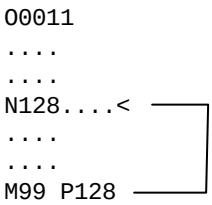
in the main program will produce an unconditional jump to the first block of the main program, and the execution of the program will be resumed there. The use of this instruction results in an endless cycle:

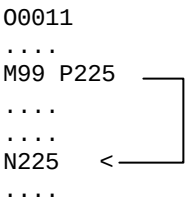


The use of instruction

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 results in an endless cycle:





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

14.1 Referring to Tool Compensation Values (H and D)

Reference can be made to

tool length compensation at address **H**,

tool radius compensation at address **D**.

The number behind the address (the tool compensation number) indicates the particular compensation value to be applied. The limit values of addresses H and D are 0 to 999.
The Table below shows the division of the compensation memory.

	Code H		Code D	
compensation number	**geometry**	**wear**	**geometry**	**wear**
01	-350.200	0.130	-32.120	0.012
02	830.500	-0.102	52.328	-0.008
.		.		.
.		.		.
.		.		.

Compensation number 00 is not included in the above Table, the compensation values pertaining to it are always zero.
Geometry value - the length/radius of the tool. It is a signed number.
Wear - the wear occurring in the course of machining. It is a signed number.
Whenever a reference is made to a compensation at address H or D in the program, the control will always take the sum of the geometry value and the wear into account for compensation. If, e.g., reference is made to H2 in the program, the length compensation will be, on the basis of the above Table, 830.500+(–0.102)=830.398.
Address H and D are modal ones, i.e., the control will take into account a given compensation value until another command D or H is given. In other words, once the compensation value has been read with command D or H, the read-out value will be unaffected by a change in the chart of compensation values (e.g., by programming G10).
The compensation values will be preserved in the memory after a power-off.
The compensation memory can be saved in the memory as a part program.

Limit values of geometry and wear:

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

The tool compensations can be selected and/or modified from the operator's panel on OFFSET screen and from the program with the use of instruction G10. If the current compensation is modified with command G10, reference has to be made again to the current compensation register D or H, or else the modified value will be disregarded.
The limit values of address H or D for the given control system, i.e., the numbers of length and radius compensations to be specified in that control system, are determined by the memory configuration of the control. In the case of a minimum memory configuration, the number of compensations is 99, i.e., the limit values of addresses H and D are 0 to 99.

14.2 Modification of Tool Compensation Values from the Program (G10)

Instruction

G10 R L P

can be used for modifying the tool compensations from the program. G10 is a one-shot instruction.
The addresses and their values have the following meanings.
The compensation value is specified at address R. At G90 (absolute data specification command), the value written at address R will be transferred to the appropriate compensation register. At G91 (incremental data specification command) or when operator I is applied, the data written at address R will be added to the content of the appropriate compensation register.
The compensation value to be modified is specified at address L:

L=10 applies to the geometry value of the length compensation (code H),
L=11 applies to the wear of the length compensation (code H),
L=12 applies to the geometry value of the radius compensation (code H)
L=13 applies to the wear of the radius compensation (code H),

The No. of compensation value to be modified is specified at address P.

L *Note*: For a programmed modification of the tool radius compensation, the value specified at address R must be interpreted as a radius in each case regardless of the state of parameter *TOOLRAD*. The control will return error message *3001 VALUE EXCESS X,Y,...F* whenever the specified values exceed the limits contained in the above Table.

14.3 Tool Length Compensation (G43, G44, G49)

Instruction

G43 *q* H or

G44 *q* H

will set up the tool length compensation mode.
Address *q* means axis *q* to which the tool length compensation is applied (*q*= X, Y, Z, U, V, W, A, B, C).
Address H means the compensation cell, from which the tool length compensation value is taken.
Irrespective of *q* being an absolute or an incremental data, instruction G43 will add the compensation value (specified at address H) to the end point coordinate obtained in the course of execution:

G43: + compensation

Irrespective of *q* being an absolute or an incremental data, instruction G44 will subtract the compensation (specified at address H) from the end point coordinate obtained in the course of execution:

G44: – compensation

Since incremental displacement Z0 has been programmed, each of instructions G43 G91 Z0 H1 and G44 G91 Z0 H1 will produce displacement just equal to the length of the tool. At G43, the displacement will be positive or negative, depending on the compensation value at H1 being positive or negative, respectively. The case is just the opposite for G44. After the command has been executed, the position displayed at coordinate Z will be the same as the one beforehand, because the position of the tool's tip will be displayed after the length compensation is set up.
Tool compensations may be defined on several axes at a time. E.g.

```
G43 Z250 H15
G43 W310 H16
```

When several axes are selected in a block, the tool length compensation will be taken into account for each axis selected:

```
G44 X120 Z250 H27
```

When the composition value is altered by calling a new H address, the previous one will be deleted, and the new value will be effective:

```
H1=10, H2=20
G90 G00
G43 Z100 H1.........moving to Z=110
G43 Z100 H2.........moving to Z=120
```

The effects of G43 and G44 are modal until another command is received from that group.

Command

G49 or

H00

will cancel the tool length compensation in each axis - with motion or with transformation, if a movement has been programmed in the block or not, respectively.
The difference between the two commands is that H00 will delete the compensation only, leaving state G43 or G44 unaffected. If a reference is made afterwards to an address H other than zero, the new tool length compensation will be set up as the function of state G43 or G44.

If, however, instruction G49 is used, any reference to address H will be ineffective until G43 or G44 is programmed.
At power-on, the value defined in parameter group *CODES* decides which code is effective (G43, G44, G49).
The example below presents a simple drilling operation with tool length compensation taken into account:
length of drilling tool, H1=400

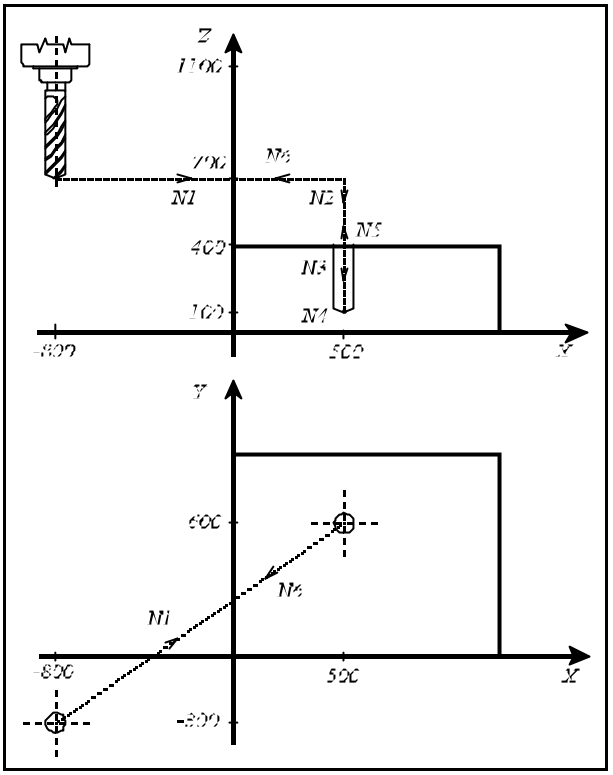


Fig. **14.3-1**

```
N1 G90 G0 X500 Y600          (positioning in plane X, Y)
N2 G43 Z410 H1               (moving to Z410 with H1 length compensation)
N3 G1 Z100 F180              (drilling as far as Z100 with F180 feed)
N4 G4 P2                     (dwell for 2 seconds)
N5 G0 Z1100 H0               (removing the tool and canceling the length
                             compensation; the tool's tip is in the X700 point)
N6 X-800 Y-300               (returning with rapid traverse in plane X, Y)
```

14.4 Tool Offset (G45...G48)

G45 increases the movement amount with the offset value
G46 decreases the movement amount with the offset value
G47 increases the movement amount by twice with the offset value
G48 decreases the movement amount by twice with the offset value

Any one of commands G45...G48 will be effective with the compensation selected with the D code until another value is called in conjunction with a command G45...G48.
Being non-modal codes, they are effective only in the block in which they have been specified.
In the case of an absolute data specification, the amount of movement will be the difference between the end point defined in the current block and the end point of the previous block. Any increase or decrease refers to the direction of motion produced in this way.

With G45 programmed (increase by the offset value):

a. movement command: 20
compensation: 5

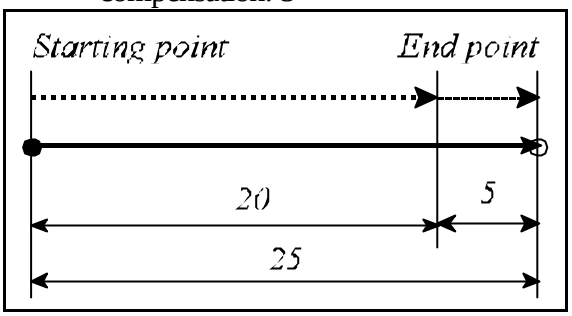


Fig. 14.4-1

b. movement command: 20
compensation: -5

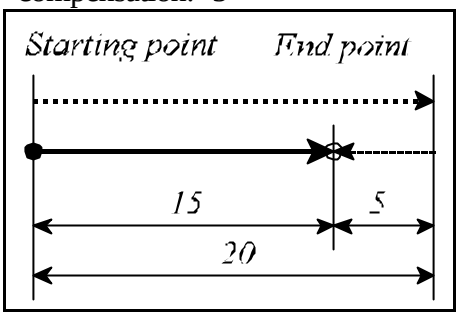


Fig. 14.4-2

a. movement command: -20
compensation: 5

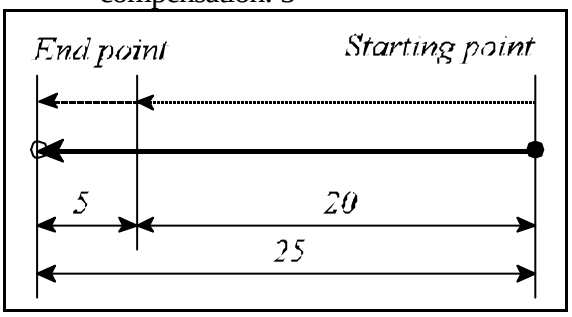


Fig. 14.4-3

b. movement command: -20
compensation: -5

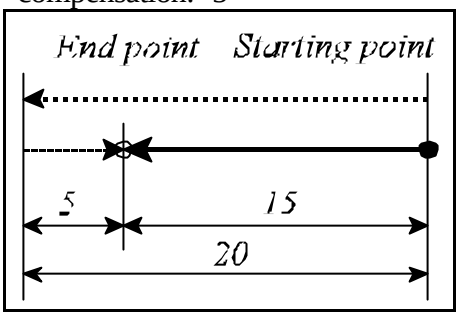


Fig. 14.4-4

With G46 programmed (decrease by the offset value):

a. movement command: 20
compensation: 5

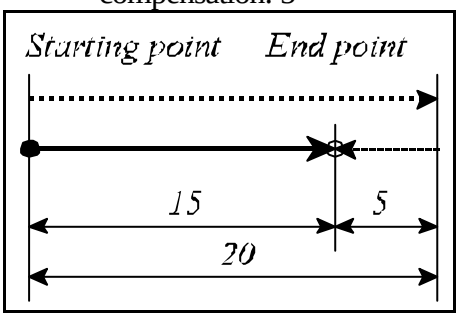


Fig. 14.4-5

cases b, c, d are similar to G45

With G47 programmed (double increase by the offset value):

a. movement command: 20 cases b, c, d are similar to G45

compensation: 5

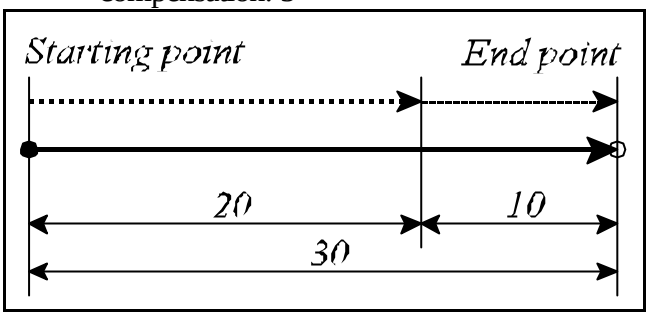


Fig. 14.4-6

With G48 programmed (double decrease by the offset value):

a. movement command: 20 cases b, c, d are similar to G45

compensation: 5

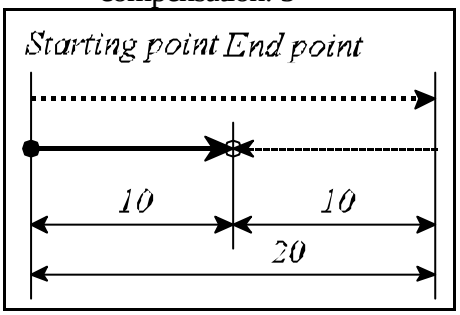


Fig. 14.4-7

If, after command G45...G48, movement commands are issued for several axes in the block, the resultant compensation will be effective in each programmed axis separately, with the value specified at D (non-vectorially generated).

If, e.g., D1=30, command G91 G45 G1 X100 Y40 D1 will produce displacements of x=130, y=70.

The resultant compensations cannot be deleted with a common G command (e.g., G49 for length compensation) or by programming D00, only with a command G45...G48 of opposite meaning.

In the use of G45...G48, only one D code may be applied, or else the control will return the error message *3008 ERRONEOUS G45...G48*.

If an incremental 0 displacement is programmed together with one of commands G45...G48, a sign preceding the 0 will also be interpreted by the control as follows:

if D1=12

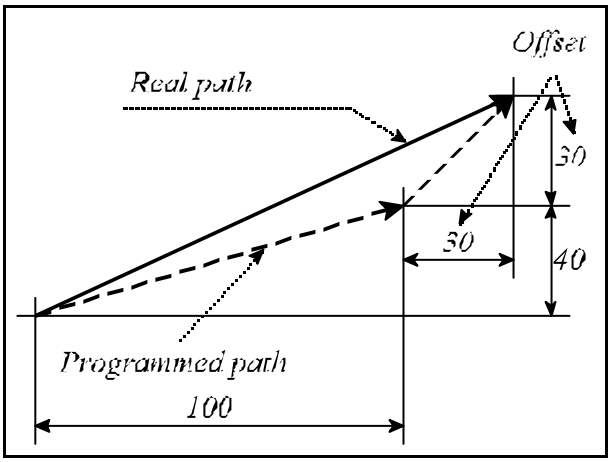


Fig. 14.4-8

NC command	G45 XI0 D1	G46 XI0 D1	G45 XI-0 D1	G46 XI-0 D1
displacement	x=12	x=-12	x=-12	x=12

A tool radius compensation applied with one of codes G45...G48 is also applicable with ¼ and ¾ circles, provided the centers of the circles are specified at address I, J or K.

An example: D1=10

```
N1 G91 G46 G0 X40 Y40 D1
N2 G47 G1 Y100 F180
N3 G47 X40
N4 Y-40
N5 G48 X60
N6 Y40
N7 G47 X20
N8 G45 Y-0
N9 G46 G3 X40 Y-40 I40
N10 G45 G1 X0
N11 G45 Y-20
N12 G45 G2 X-40 Y-40 I-40
N13 G45 G1 X-120
N14 G46 G0 X-40 Y-40
```

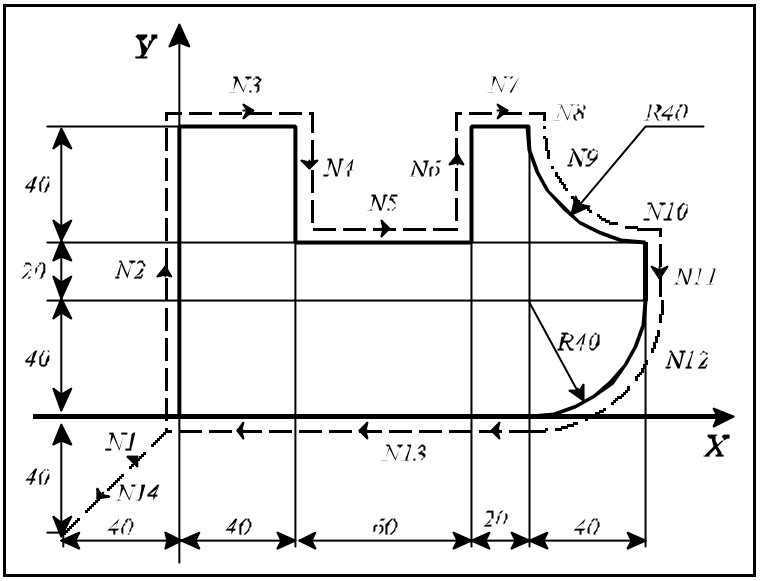


Fig. 14.4-9

14.5 Cutter Compensation (G38, G39, G40, G41, G42)

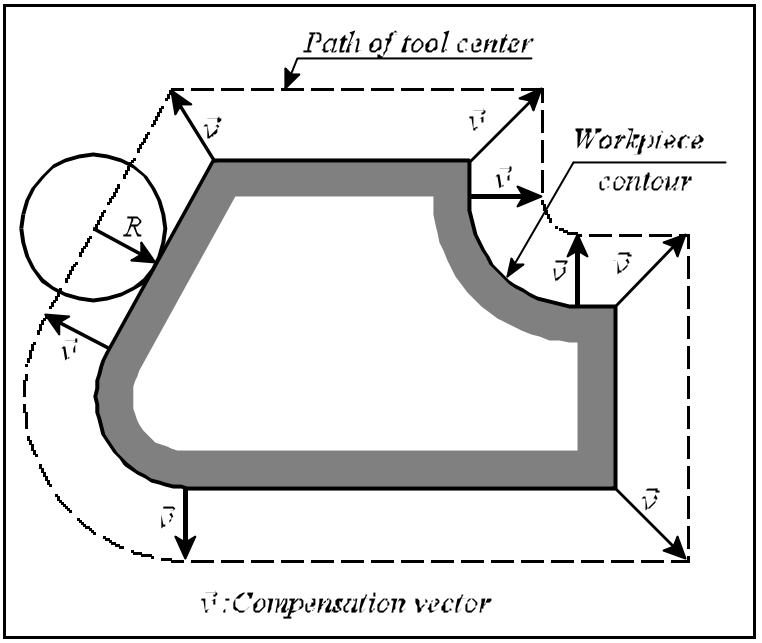


Fig. 14.5-1

To be able to mill the contour of a two-dimensional workpiece and to specify the points of that formation as per the drawing in the program (regardless of the size of the tool employed), the control must guide the tool center parallel to the programmed contour, spaced by a tool radius from the latter. The control will determine the distance between the path of the tool center and the programmed contour in accordance with the compensation value of the tool radius referred to by compensation number D.
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 D) 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 cutter compensation. Movements outside of this plane are not influenced by compensation. If, e.g., plane X, Y is selected in state G17, the compensation vectors will be computed in that plane. In this case any movement in Z 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: Cutter compensation cancel
G41: Cutter compensation left
G42: Cutter compensation right

Command G41 or G42 will set up the compensation computation. In state G41 or G42 the programmed 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 address D. The specification of D00 is always equivalent to calling zero radius value. The

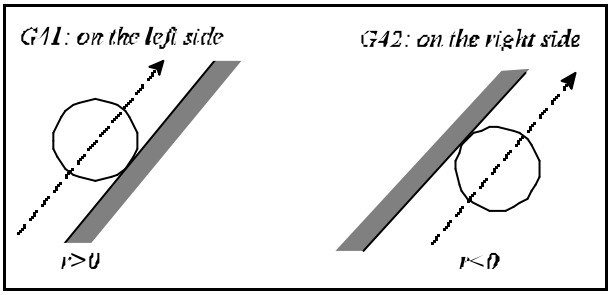


Fig. 14.5-2

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 mill 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

L *Note*:

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

Command G40 or D00 will cancel the offset compensation. The difference between the two commands is that D00 deletes only the compensation vector, leaving state G41 or G42 unchanged. If a reference is made subsequently to an address D other than zero, 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, any reference to address D will be ineffective until G41 or G42 has been programmed.
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.

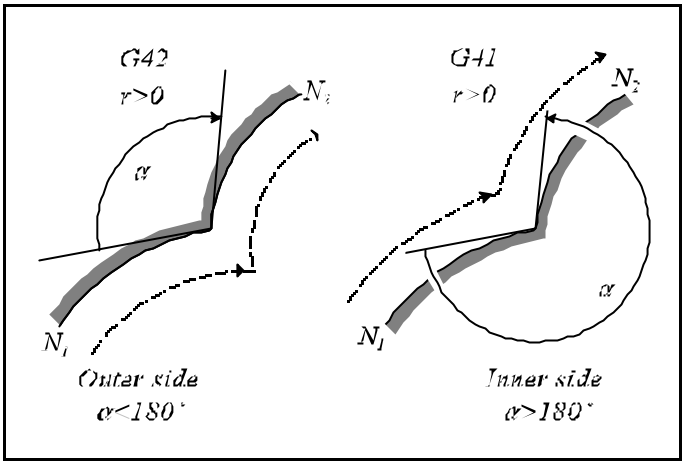


Fig. 14.5-3

An auxiliary data is to be introduced before embarking on the discussion of the details of the compensation computation. It is "", the angle at the corner of two consecutive blocks viewing from the workpiece side. The direction of " 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 ". If ">180°, i.e., the tool is working inside, the control will compute a point of intersection between the two interpolations. If "<180°, i.e., the tool is moving around the outside, it *may add* further straight sections.

14.5.1 Start up of Cutter 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 tool center 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. The value of compensation will be taken from the compensation cell (D register). 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 cutter 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 D00 and re-activated with Dnn (nn being a number other than 0).
The basic instances of starting compensation up depending on the angle of " 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.

L *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 center,
Continuos line: the programmed path.

Basic instances of starting up the cutter compensation:

(G40)	(G40)
G42 G01 X_ Y_ D_	G42 G01 X_ Y_ D_
X_ Y_	G2 X_ Y_ R_

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

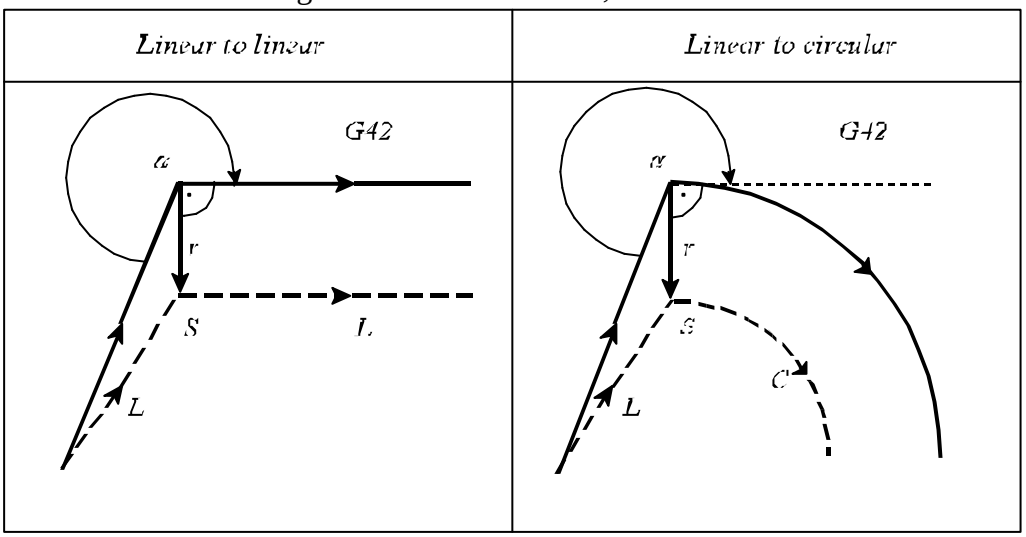


Fig. **14.5.1-1**

Going around the outside of a corner at an obtuse angle, 90°#"#180°

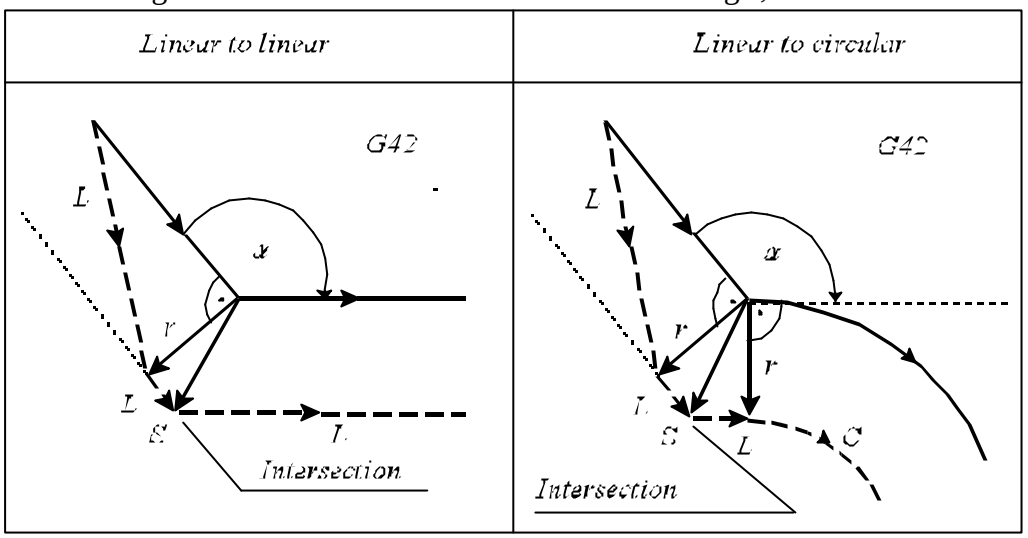


Fig. 14.5.1-2

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

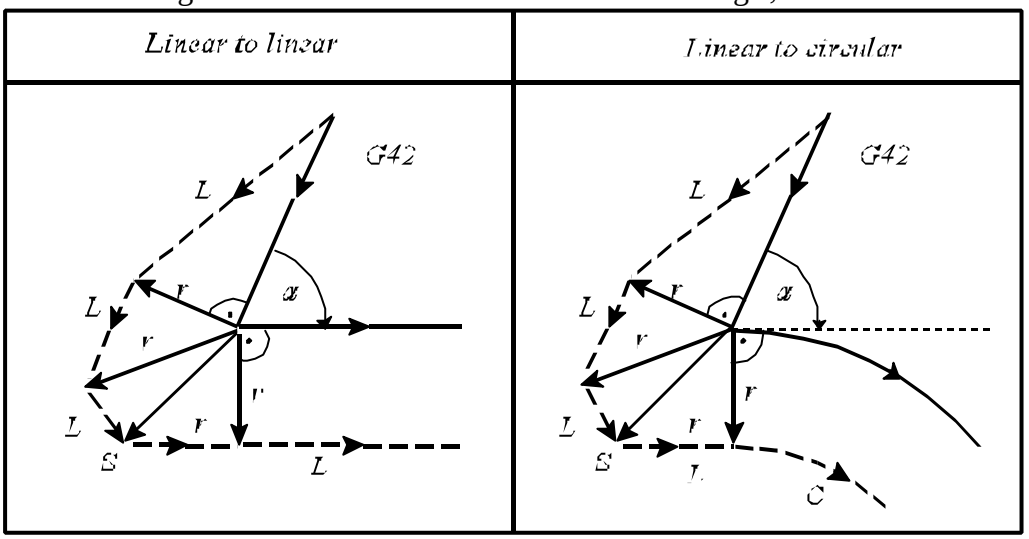


Fig. 14.5.1-3

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, J in the case of G17) - 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.

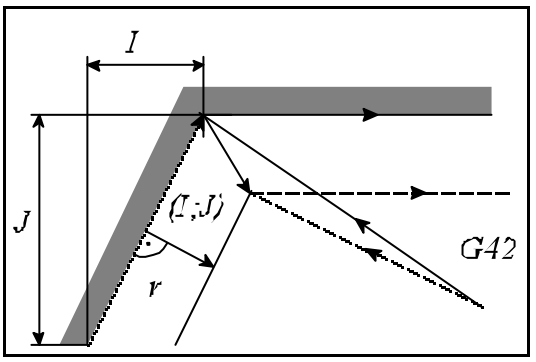


Fig. 14.5.1-4

```
...
G91 G17 G40
...
N110 G42 G1 X-80 Y60 I50 J70 D1
N120 X100
...
```

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.

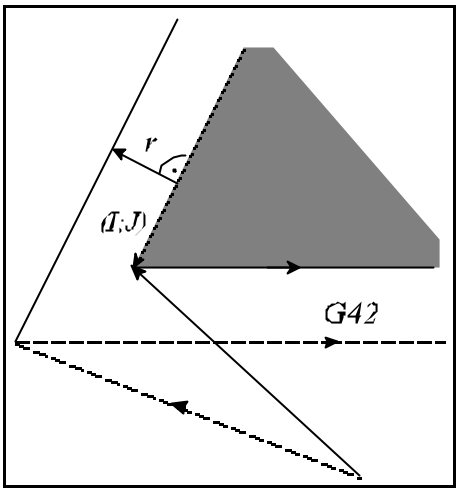


Fig. 14.5.1-5

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

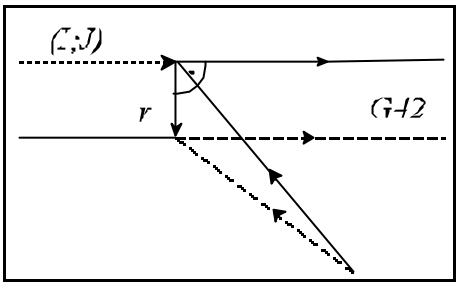


Fig. 14.5.1-6

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 G17 G0 X0 Y0
N15 G42 D1
N20 G1 X80
N25 X110 Y60
...
```

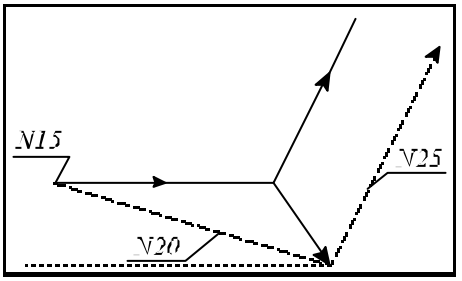


Fig. 14.5.1-7

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 G17 G0 X0 Y0
N15 G91 G42 D1 X0
N20 G1 X80
N25 X30 Y60
...
```

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 G17 G0 X0 Y0
N15 G91 G42 D1 X80
N20 G1 X0
N25 X30 Y60
N30 X60
...
```

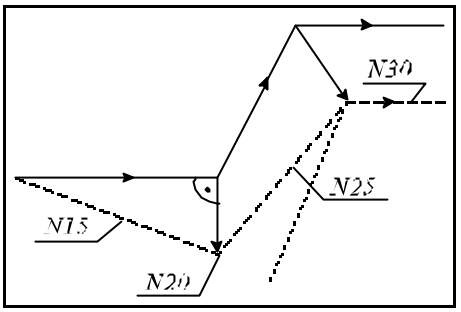


Fig. **14.5.1-8**

14.5.2 Rules of Cutter 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°<"<360°

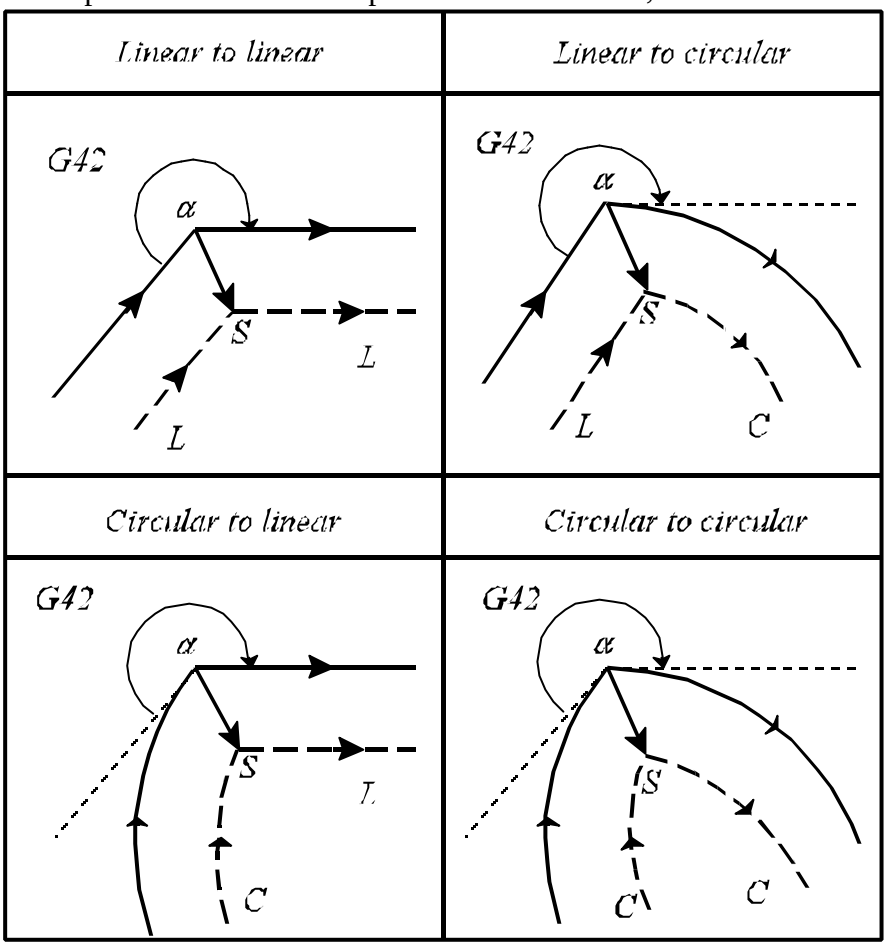


Fig. 14.5.2-1

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

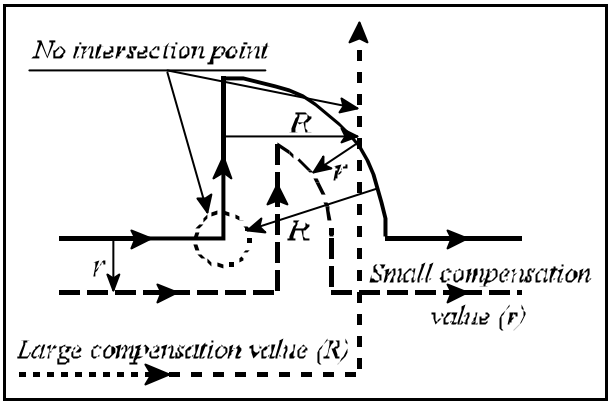


Fig. 14.5.2-2

Going around the outside of a corner at an obtuse angle, 90°#"#180°

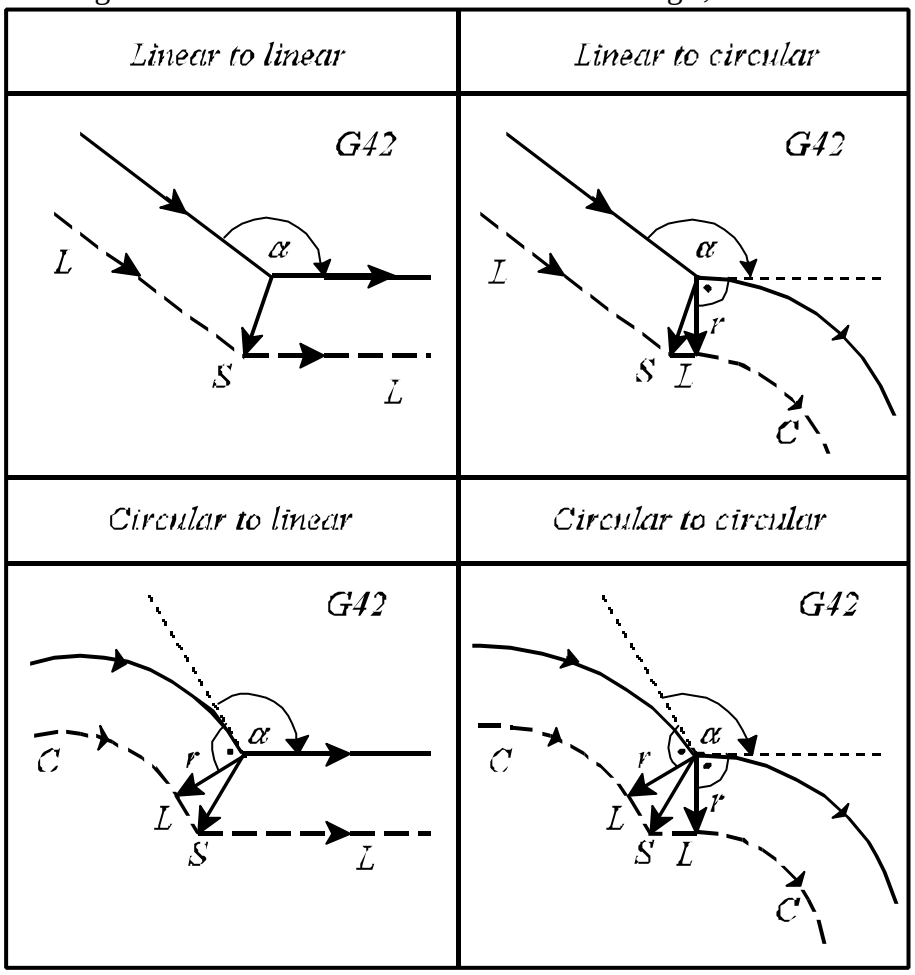


Fig. 14.5.2-3

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

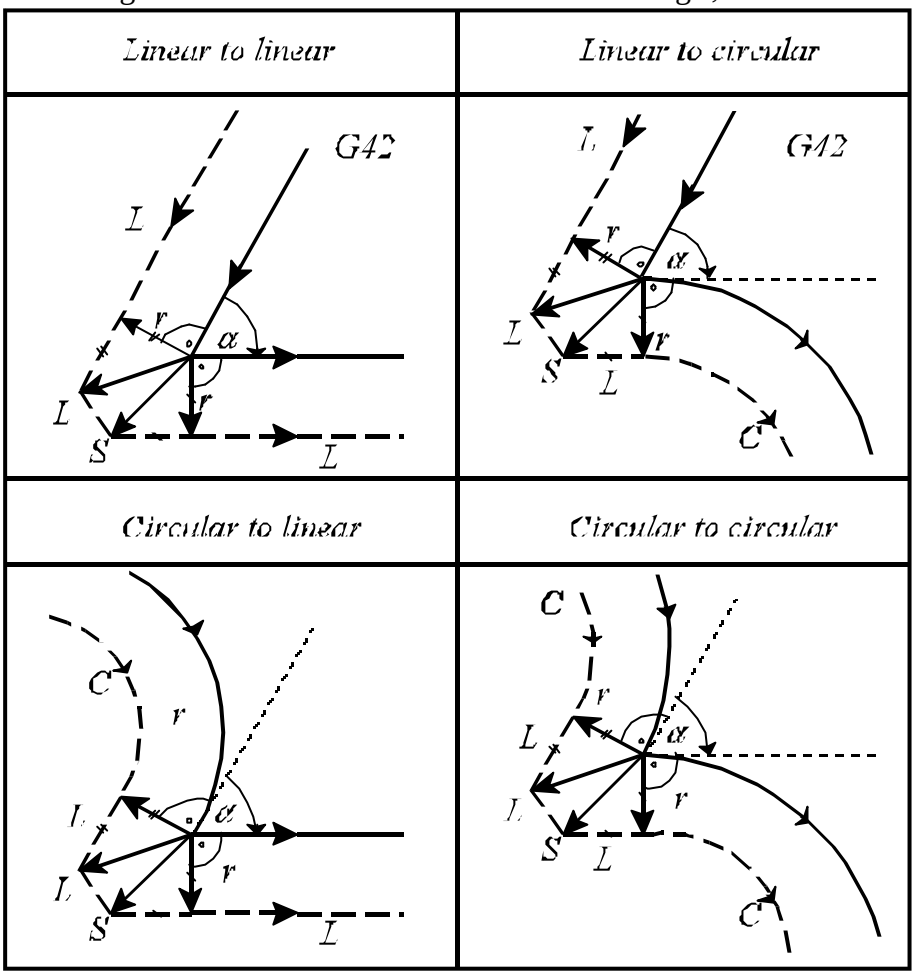


Fig. 14.5.2-4

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 compensation. 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 G17 G42...
N110 G1 X40 Y50
N120 X0
N130 X90
N140 X50 Y-20
...
```

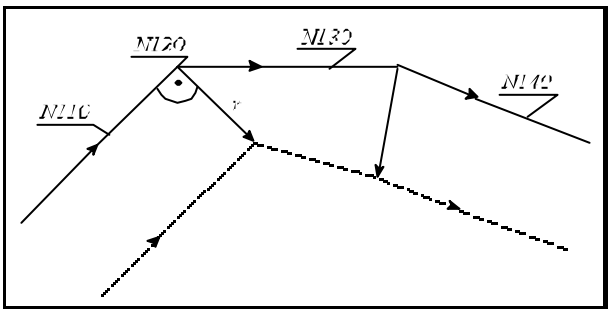


Fig. 14.5.2-5

14.5.3 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_ Y_	G02 X_ Y_ R_
G40 X_ Y_	G40 G1 X_ Y_

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

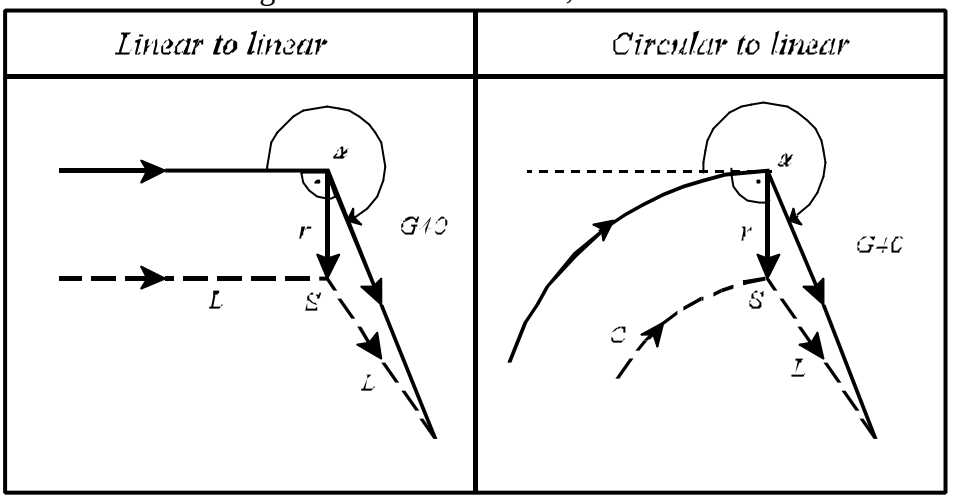


Fig. 14.5.3-1

Going around the outside of a corner at an obtuse angle, 90°#"#180°

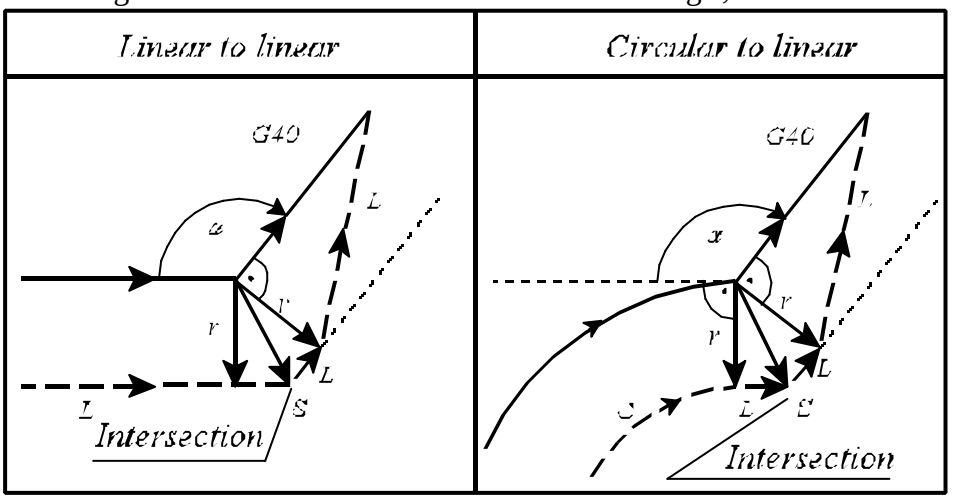


Fig. 14.5.3-2

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

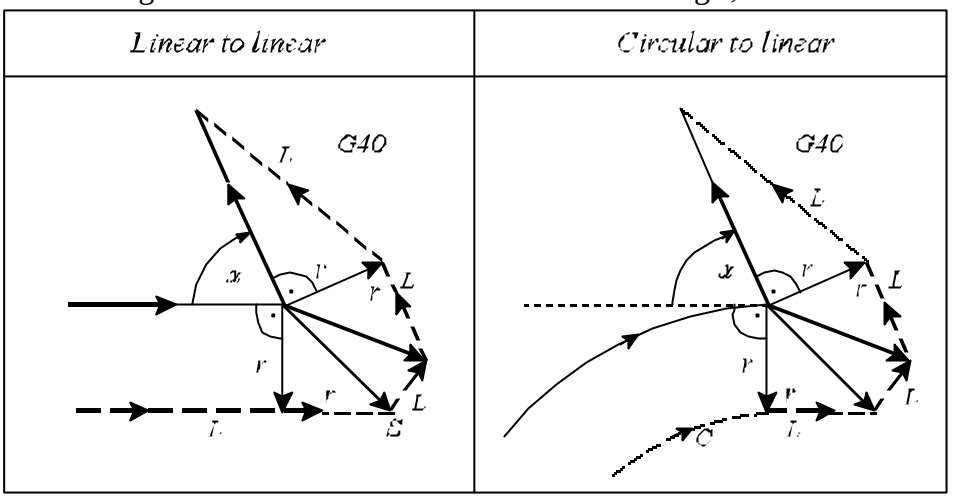


Fig. **14.5.3-3**

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, J in the case of G17) - 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 G17 G42...
N100 G1 X50 Y60
N110 G40 X70 Y-60 I100 J-20
...
```

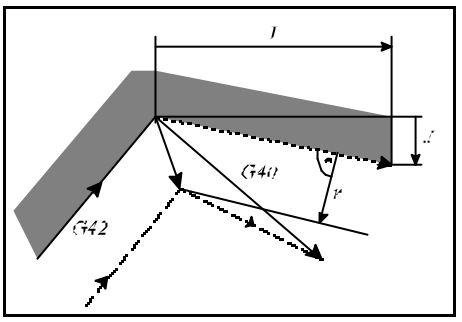


Fig. **14.5.3-4**

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.

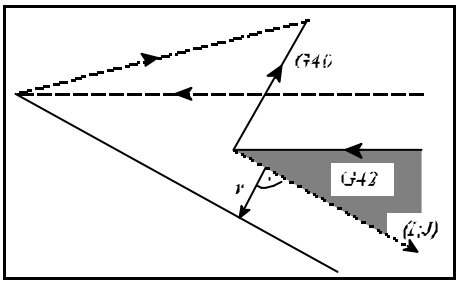


Fig. **14.5.3-5**

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

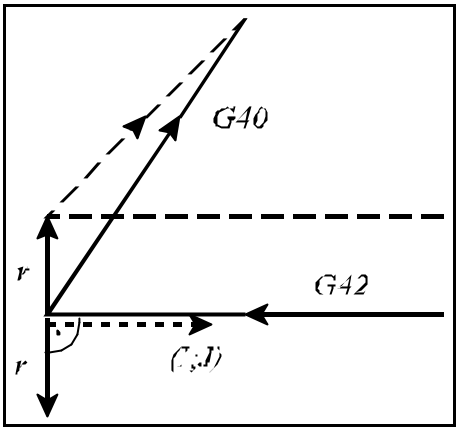


Fig. 14.5.3-6

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 G17 G91...
N110 G1 X80 Y40
N120 G40
N130 X-70 Y20
...
```

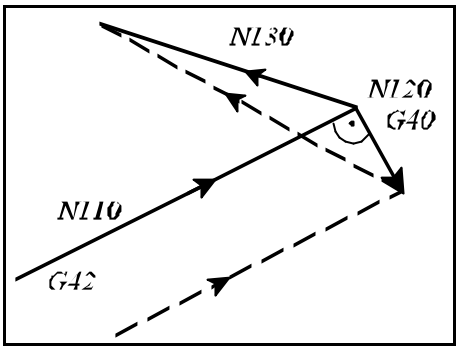


Fig. 14.5.3-7

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 G17 G91...
N110 G1 X80 Y40
N120 G40 X0
N130 X-70 Y20
...
```

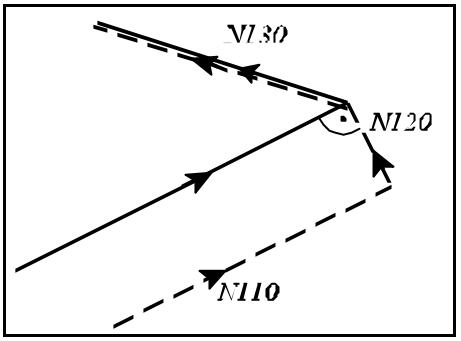


Fig. 14.5.3-8

14.5.4 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.

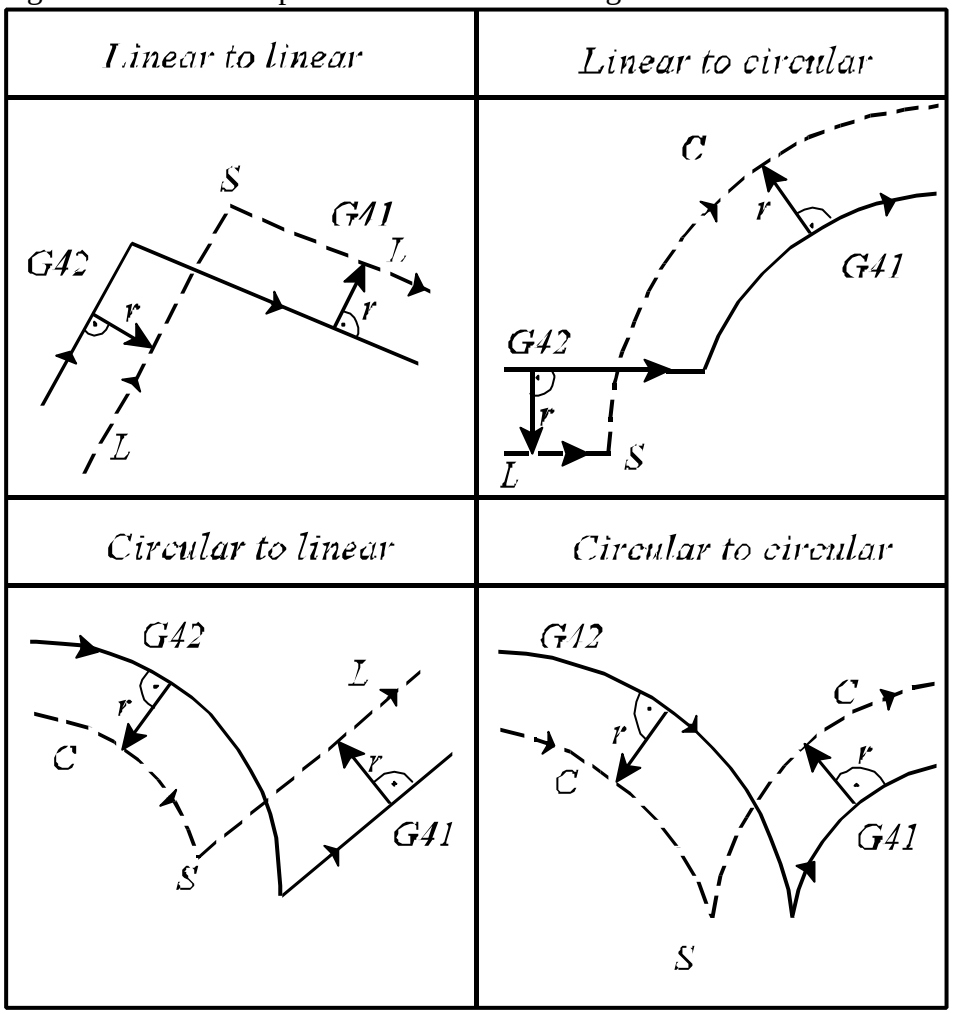


Fig. **14.5.4-1**

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

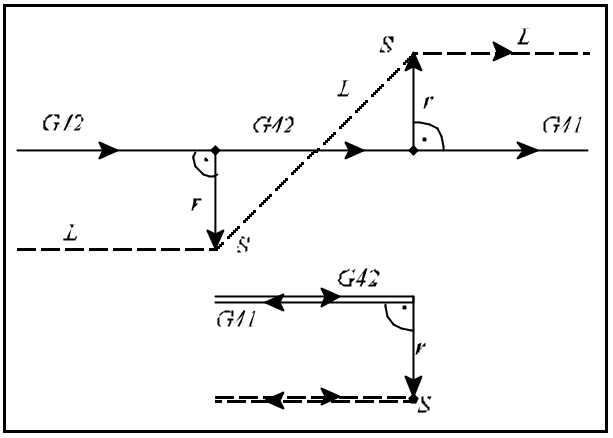


Fig. 14.5.4-2

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

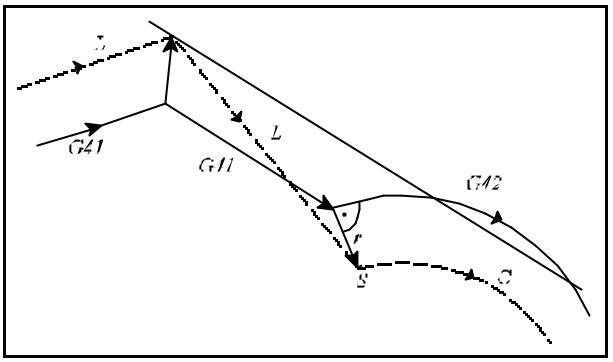


Fig. 14.5.4-3

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.

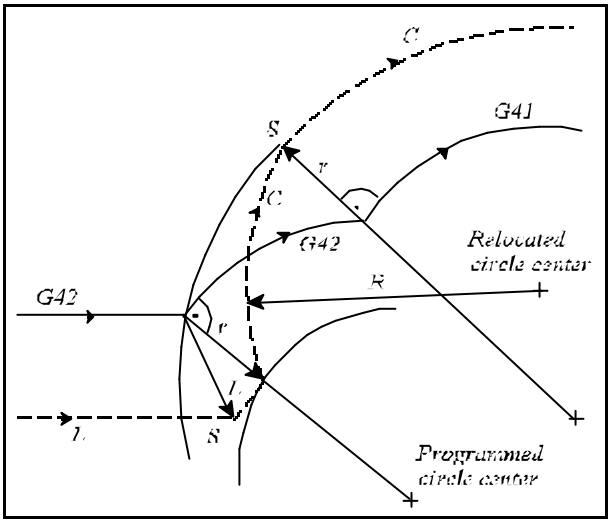


Fig. 14.5.4-4

14.5.5 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:

```
...G17 G41 G91...
N110 G1 X60 Y60
N120 G38 X90 Y-40
N130 X20 Y70
N140 X60
...
```

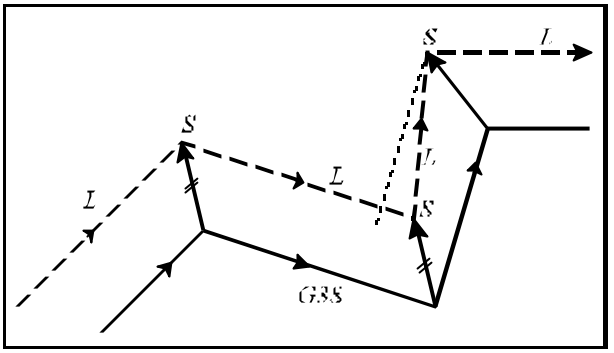


Fig. **14.5.5-1**

To program a recession without canceling the offset mode:

```
...G17 G42 G91...
N110 G1 X40
N120 G38 X50
N130 G38 Y70
N140 G38 Y-70
N150 X60
...
```

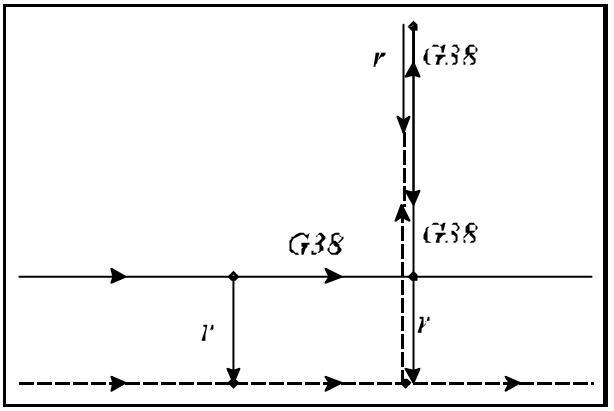


Fig. **14.5.5-2**

14.5.6 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:

```
...G17 G91 G41...
N110 G1 X100
N120 G39
N130 G3 X80 Y-80 I80
...
```

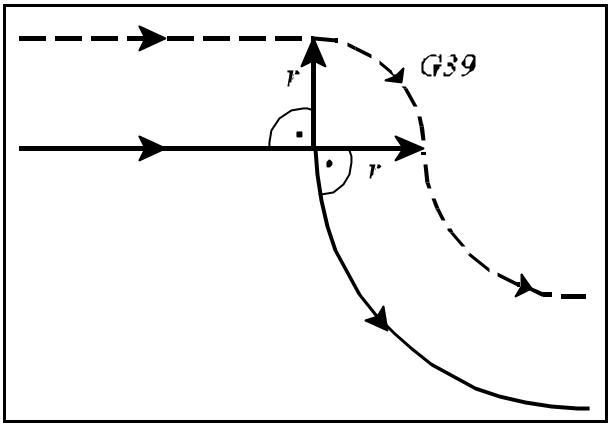


Fig. **14.5.6-1**

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.

```
...G17 G91 G41...
N110 G1 X100
N120 G39 I50 J-60
N130 G40 X110 Y30
...
```

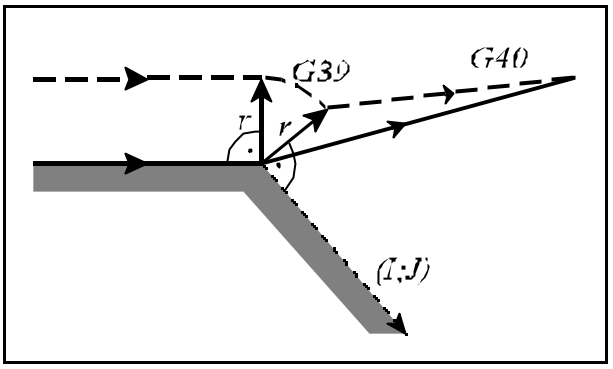


Fig. **14.5.6-2**

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.

14.5.7 General Information on the Application of Cutter 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 ([G17] G1 Z)
– call of a subprogram (M98 P)
– setting or canceling special transformations (G50, G51, G50.1, G51.1, G68, G69).

L *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:

```
...G17 G42 G91...
N110 G1 X50 Y70
N120 G4 P2
N130 X60
...
```

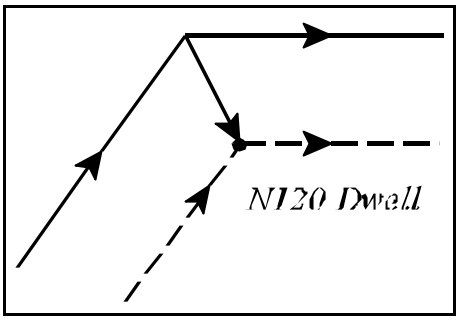


Fig. 14.5.7-1

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:

```
...G17 G42 G91...
N110 G1 X50 Y70
N120 G4 P2
N130 S400
N140 X60
...
```

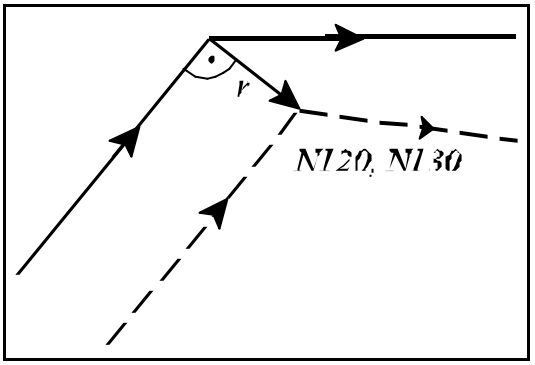


Fig. 14.5.7-2

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

```
...G17 G91...
N110 G41 G0 X50 Y70 D1
N120 G1 Z-40
N130 Y40
...
```

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

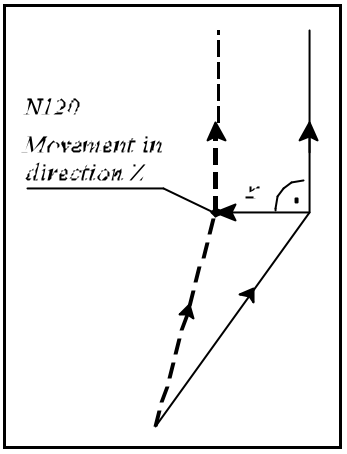


Fig. 14.5.7-3

If, however, movement in direction **Z** 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:

```
...G17 G91...
N110 G41 G0 X50 Y70 D1
N120 Z-35
N130 G1 Z-5
N140 Y40
...
```

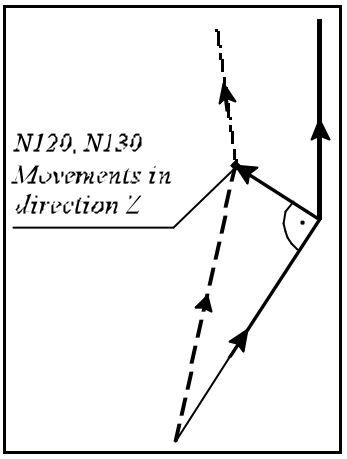


Fig. 14.5.7-4

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

```
...G17 G91...
N110 G41 G0 X50 Y69 D1
N120 Z-35
N130 Y1
N140 G1 Z-5
N150 Y40
...
```

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

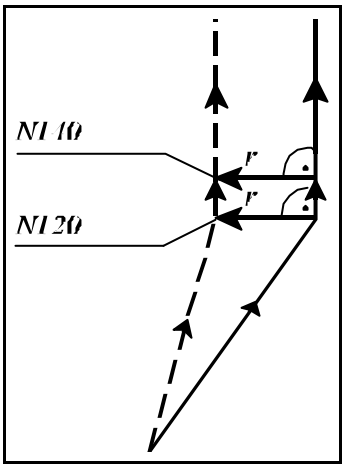


Fig. 14.5.7-5

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 G17
G41...
N110 G1 X80 Y-50
N120 G92 X0 Y0
N130 X80 Y50
...
```

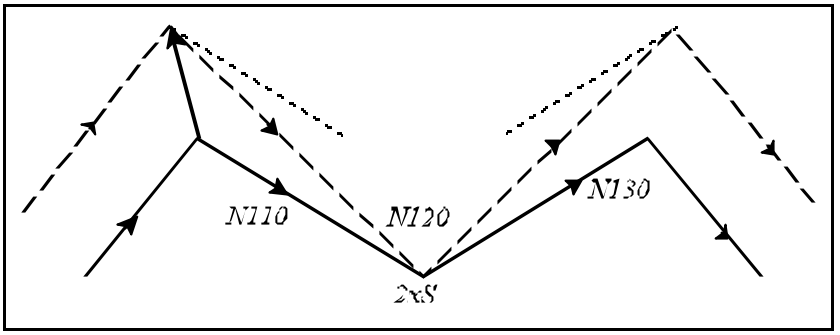


Fig. **14.5.7-6**

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 G17
G41...
N110 G1 X80 Y-50
N120 G53 Y80
N130 G53 Y0
N140 X80 Y50
...
```

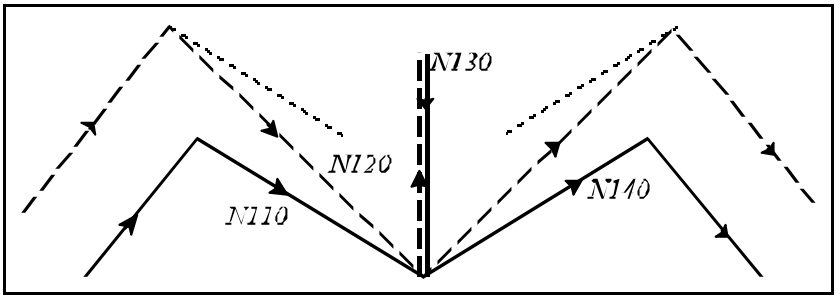


Fig. **14.5.7-7**

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 G17
G41...
N110 G1 X80 Y-50
N120 G28 Y80
N130 G29 Y0
N140 X80 Y50
...
```

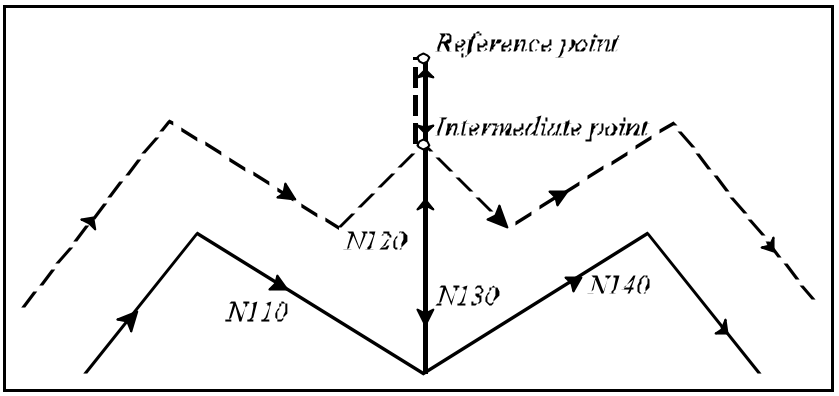


Fig. 14.5.7-8

A new compensation value can also be called at address D 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 D 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.

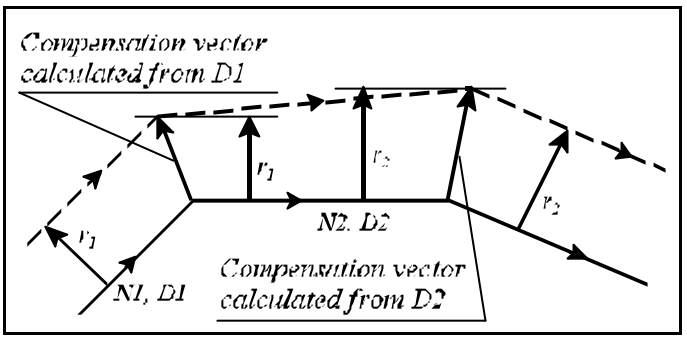


Fig. 14.5.7-9

A new radius compensation value can be called at address D in a circular interpolation, too, this time, however, the tool center will be moving along an arc with a variable radius.

A special instance of the foregoing is canceling or setting up the compensation with D00 or Dnn, 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 D.

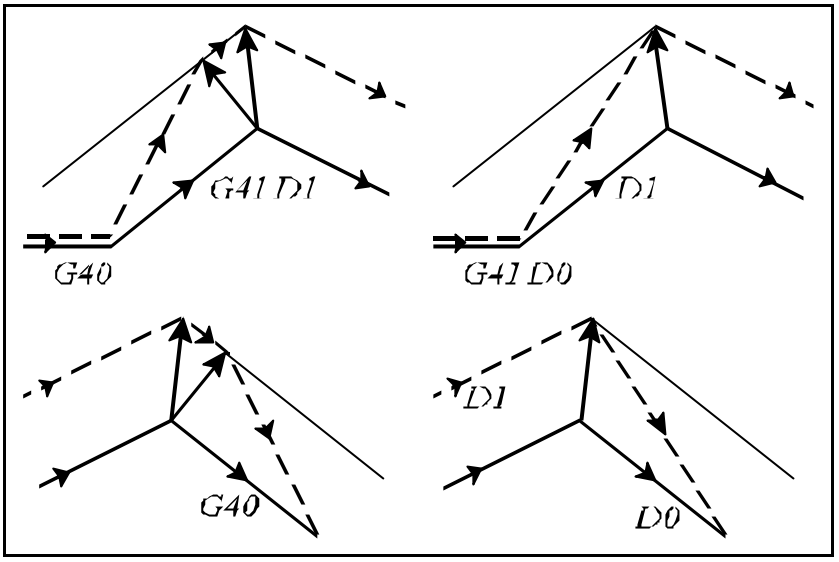


Fig. 14.5.7-10

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 X80 D1
N030 G1 Z-5
N040 G3 I-80
N050 G1 Z2
N060 G40 G0 X0
...
```

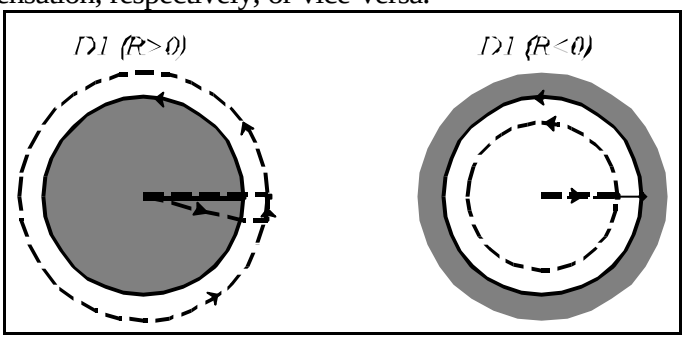


Fig. 14.5.7-11

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.

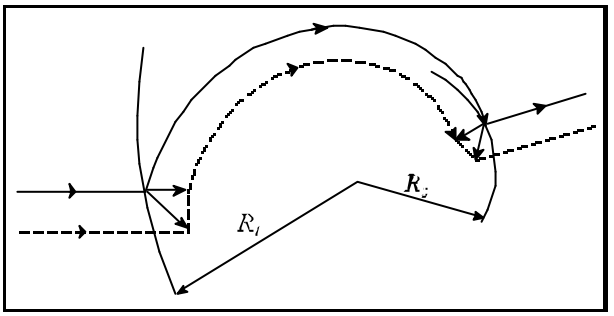


Fig. 14.5.7-12

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.

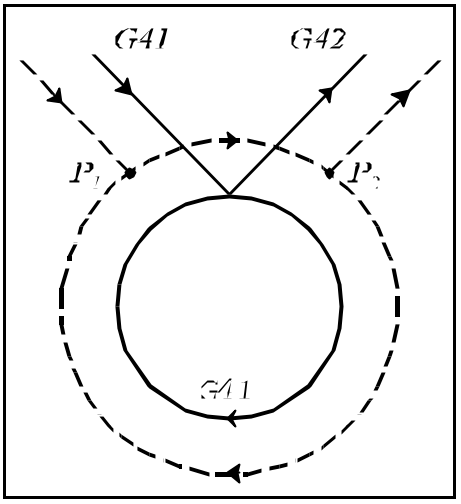


Fig. 14.5.7-13

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

```
...G17 G90 G41...
N090 G1 X30
N100 G2 J-60
N110 G40 G1 X120 Y180 I-60 J-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.

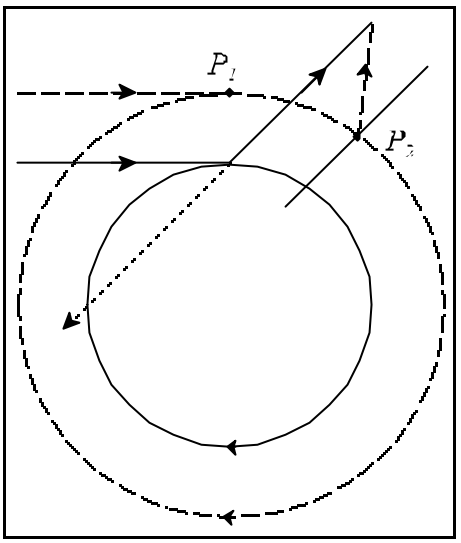


Fig. 14.5.7-14

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.

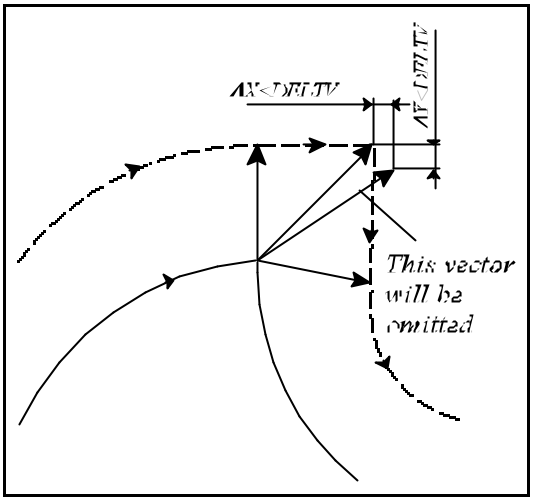


Fig. 14.5.7-15

14.5.8 Interferences in Cutter 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 cutter 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.

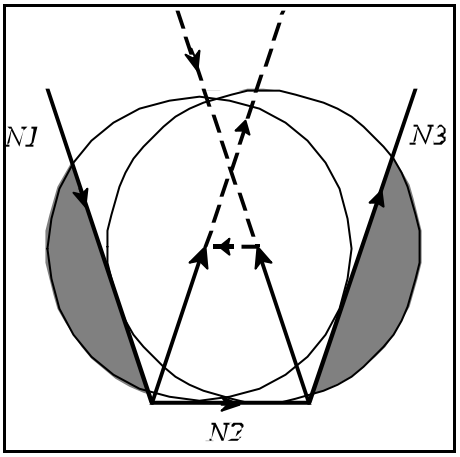


Fig. 14.5.8-1

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°#n#+90° is fulfilled for angle n between the programmed displacement and the one compensated with the radius.

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

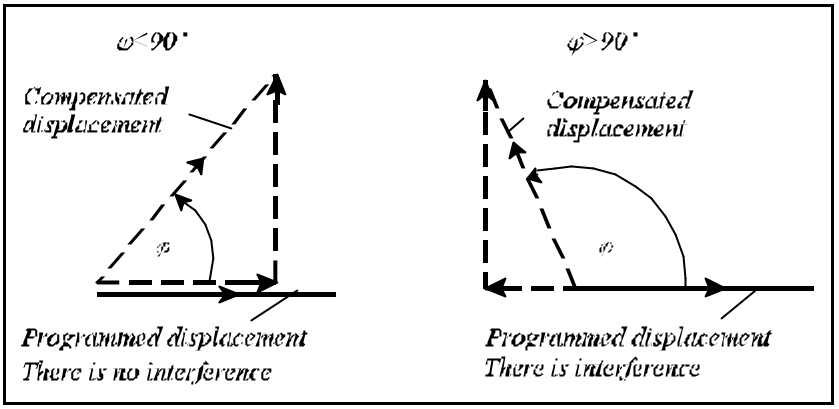


Fig. 14.5.8-2

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.

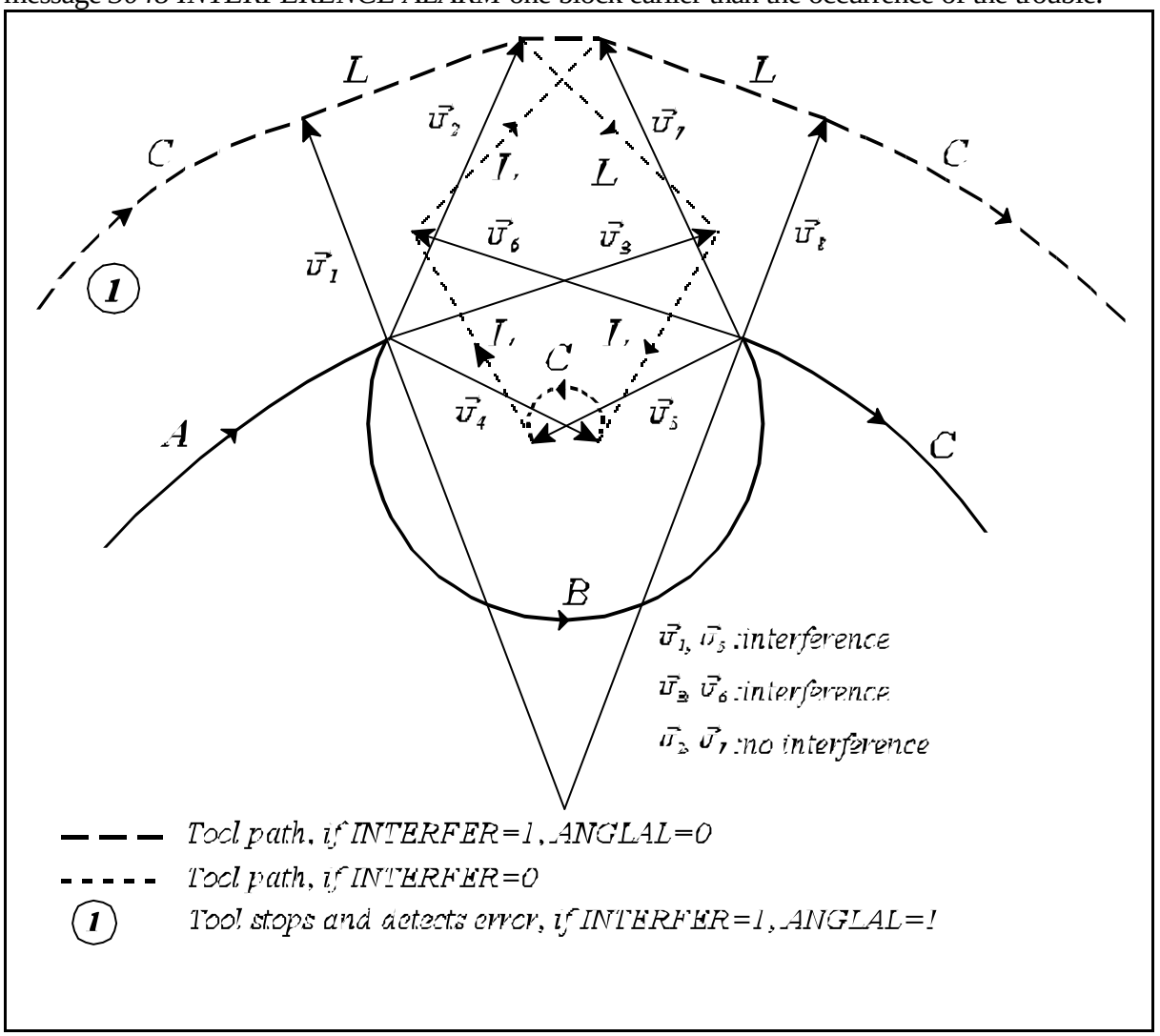


Fig. 14.5.8-3

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 $\vec{P}_1, \vec{P}_2, \vec{P}_3, \vec{P}_4$; the compensation vectors between blocks B and C are $\vec{P}_5, \vec{P}_6, \vec{P}_7, \vec{P}_8$.

- $\vec{P}_4$ and $\vec{P}_5$ will be ignored if there is an interference between them.
- $\vec{P}_3$ and $\vec{P}_6$ will be ignored if there is an interference between them.
- $\vec{P}_2$ and $\vec{P}_7$ will be ignored if there is an interference between them.
- $\vec{P}_1$ and $\vec{P}_8$ 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.

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

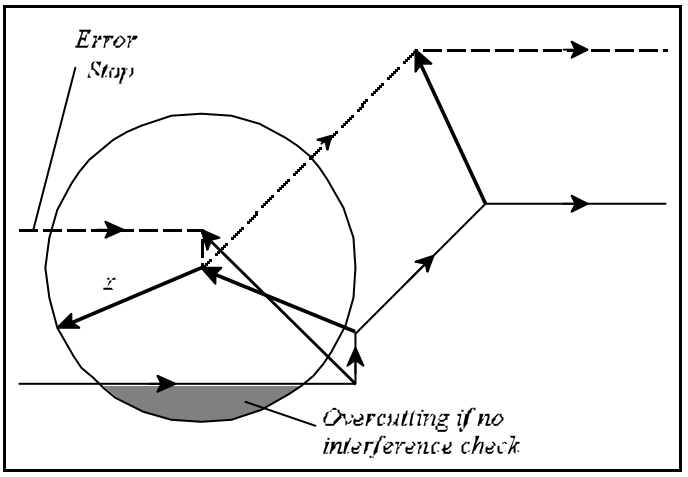


Fig. **14.5.8-4**

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

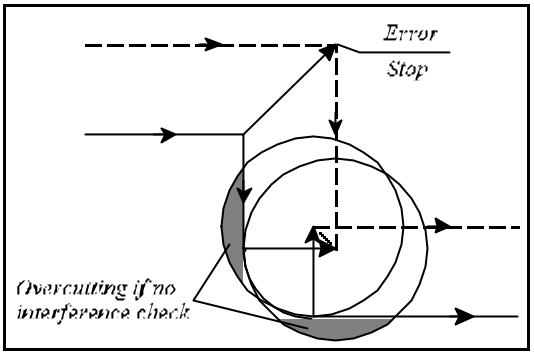


Fig. 14.5.8-5

Milling a step smaller than the tool radius along an arc. If parameter *ANGLAL* is 0, the control will delete vector $\vec{P}_2$ and will interconnect vectors $\vec{P}_1$ and $\vec{P}_3$ 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.

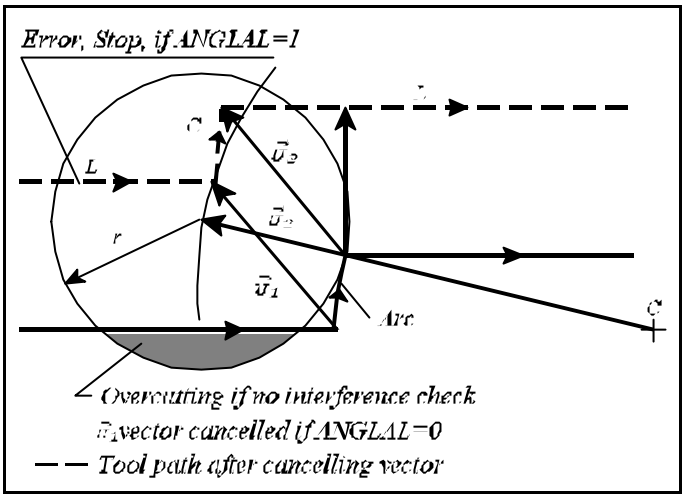


Fig. 14.5.8-6

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.

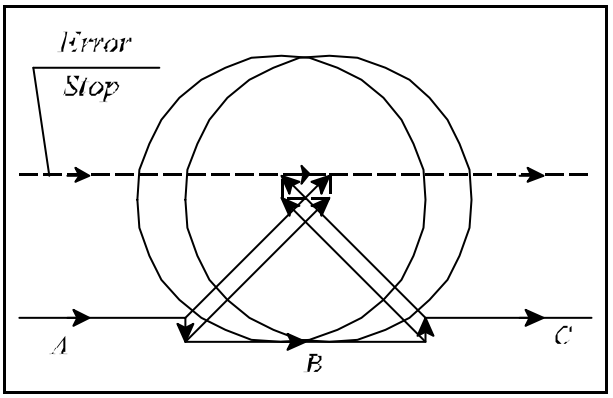


Fig. 14.5.8-7

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.

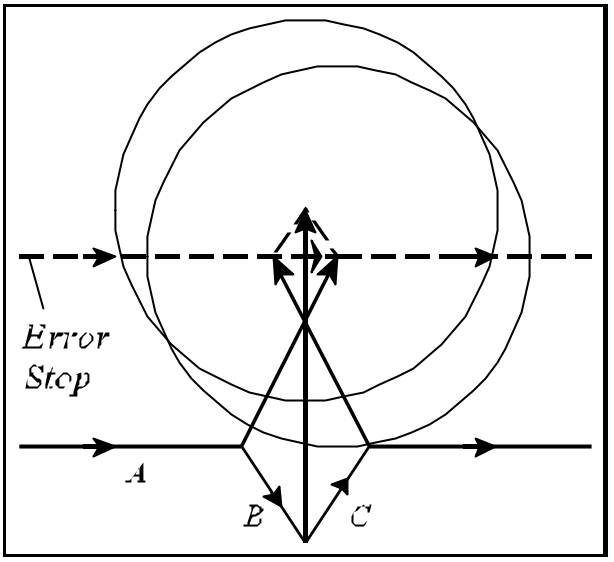


Fig. 14.5.8-8

14.6 Three-dimensional Tool Offset (G41, G42)

The 2D tool radius compensation will offset the tool in the plane selected by commands G17, G18, G19. The application of the three-dimensional tool compensation enables the tool compensation to be taken into account in three dimensions.

14.6.1 Programming the Three-dimensional Tool Offset (G40, G41, G42)

Command

G41 (G42) X_p Y_p Z_p I J K D (E)

will set up the 3D tool compensation.

X_p, Y_p, Z_p mean axes X, Y, Z or axes parallel to them (if any).

Unless reference is made to an axis the principal axes will be taken into account automatically. For example,

instruction G41 X I J K refers to space X Y Z

instruction G41 U V Z I J K refers to space U V Z

instruction G41 W I J K refers to space X Y W.

When the three-dimensional tool compensation is set up, each of addresses I, J, K has to be specified, or else the control will assume the state of 2D tool-radius compensation.

The values specified at addresses I, J, K are the components of the three-dimensional compensation vector. The values of the components are modal, i.e., each will remain effective until a reference is made to another value of I, J or K.

The compensation value to be applied can be called at address D.

The dominator constant of compensation calculation can be specified at address E.

Command

G40 or

D00

will cancel the three-dimensional offset compensation.
The difference between the two commands is that D00 will delete the compensation only, leaving state G41 or G42 unchanged. If a reference is made subsequently to a new address D (other than zero), the new tool compensation will be set up as the function of state G41 or G42.
If, however, instruction G40 is used, any reference to address D will be ineffective until G41 or G42 is programmed.
The compensation computation can be set up (G41, G42) or canceled (G40 or D00) only in a block of linear movement (G00 or G01).
G40, G41, G42 are modal ones. The control will assume state G40 after power-on.

14.6.2 The Three-dimensional Offset Vector

The control will generate the components of compensation vectors in the following way:

$$v_x = \frac{I * r}{P}$$
$$v_y = \frac{J * r}{P}$$
$$v_z = \frac{K * r}{P}$$

where r is the compensation value called at address D,
P is the dominator constant,
I, J, K are values specified in the program.

The value of dominator constant is taken from parameter *DOMCONST* unless a different value is specified in the program at address E. If the value of the dominator constant is 0 and no value has been specified at address E either, the control will compute the value of P from the relationship

$$P = \sqrt{I^2 + J^2 + K^2}$$

Based on the directions of compensation vectors specified in each block, the control will take the compensations into account in block after block. Thus, in the course of a three-dimensional machining, the CAM system need not generate the path to a given tool, instead, only the vectorial directions have to be computed at the end points of the interpolations. Then the programs generated in this way can be run with the use of tools of different dimensions as well.

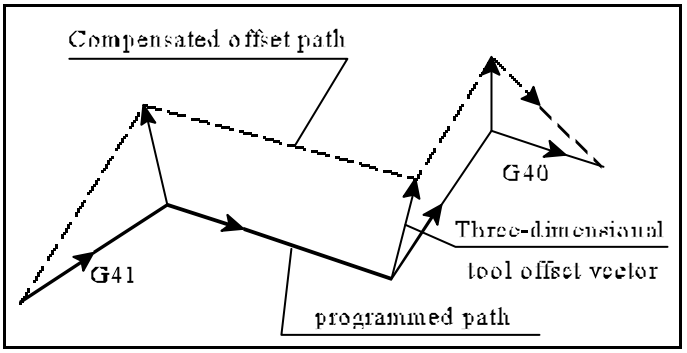


Fig. 14.6.2-1

The compensation vector cannot be altered in a circular interpolation, i.e., the compensation vectors are identical at the beginning and end of a circular interpolation.

Instruction G42 functions in the same manner as G41 with the difference that the compensation vector is computed in a direction opposite to G41:

$$v_x = -\frac{I * r}{P}$$
$$v_y = -\frac{J * r}{P}$$
$$v_z = -\frac{K * r}{P}$$

A change-over from state G41 to G42 or vice versa is only feasible in a linear interpolation block. The previous values will be modal if - with the three-dimensional tool compensation set up - I and J and K are all omitted in an interpolation.
It is not feasible to set up the three-dimensional compensation and two-dimensional radius compensation simultaneously.

15 Special Transformations

15.1 Coordinate System Rotation (G68, G69)

A programmed shape can be rotated in the plane selected by G17, G18, G19 by the use of command

G68 p q R

The coordinates of the center of rotation will be specified at address p and q. The system will only interpret the data written at coordinates p and q of the selected plane.
The entered p and q coordinate data are also interpreted as rectangular coordinate data even when polar coordinate data specifications are set up. Using G90, G91 or operator I, the p and q coordinates of the center of rotation can be specified as absolute or incremental data.
Unless one or both of p and q are assigned values, the instantaneous axis position will be taken for the center of rotation.

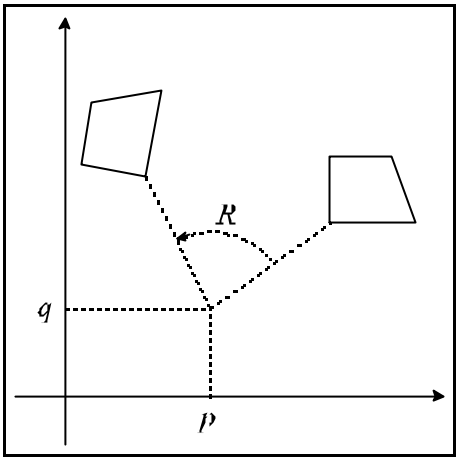


Fig. 15.1-1

The angle of rotation is specified at address R. A positive or negative value written at the address R represents a counter-clockwise or a clockwise direction of rotation, respectively.
The value of R can be specified in 8 decimal digits. The accuracy of rotation can be selected with reference to parameter *ANG.ACCU*. If its value is 0 or 1, the accuracy of calculating the rotation will be 0.001° or 0.00001°, respectively.
The value specified for R may be absolute or incremental. When the angle of rotation is specified as an incremental data, the value of R will be added to the previously programmed angles.

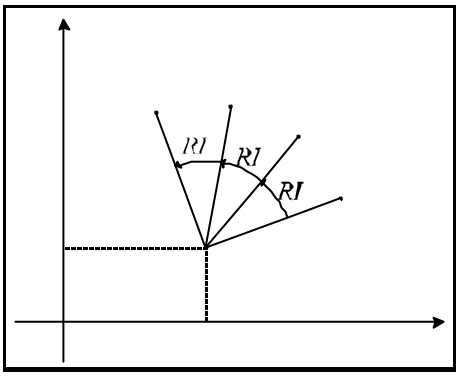


Fig. 15.1-2

The rotation can be canceled with command

G69.

The coordinates of the center of rotation and the angle will be deleted. That instruction may accompany other commands as well.

Example:

```
N1 G17 G90 G0 X0 Y0
N2 G68 X90 Y60 R60
N3 G1 X60 Y20 F150
   (G91 X60 Y20 F150)
N4 G91 X80
N5 G3 Y60 R100
N6 G1 X-80
N7 Y-60
N8 G69 G90 X0 Y0
```

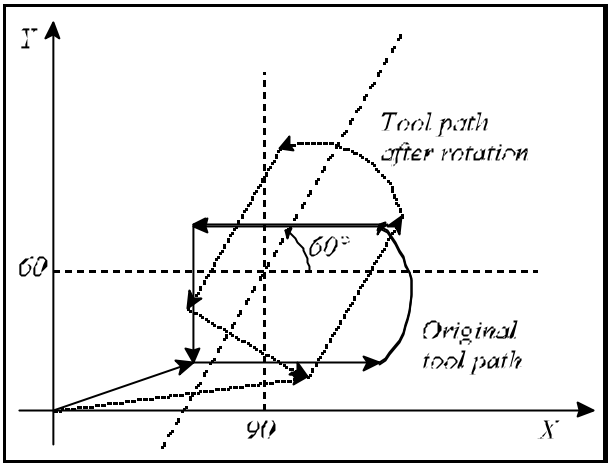


Fig. 15.1-3

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

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.

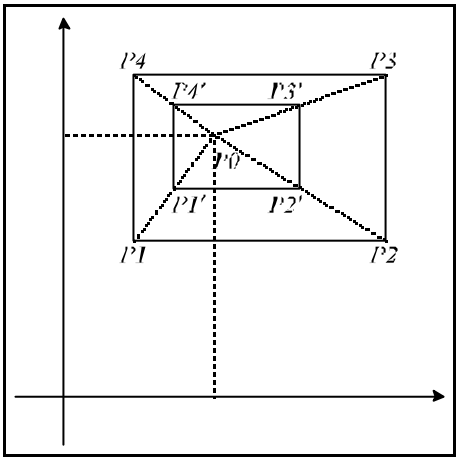


Fig. 15.2-1

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 X0 Y0
N2 G51 X60 Y140 P0.5
N3 G1 X30 Y100 F150
   (G91 X30 Y100 F150)
N4 G91 X100
N5 G3 Y60 R100
N6 G1 X-100
N7 Y-60
N8 G50 G90 X0 Y0
```

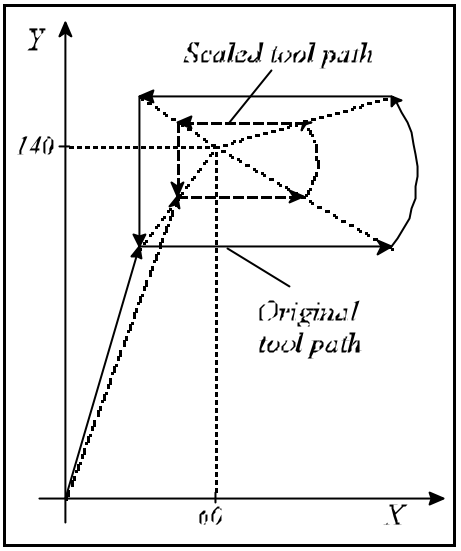


Fig. 15.2-2

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:

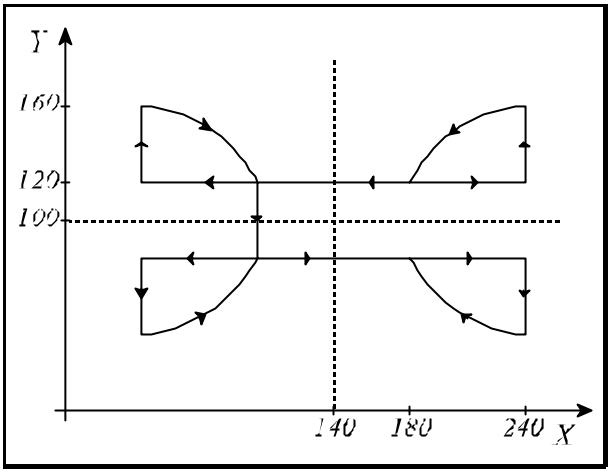


Fig. 15.3-1

subprogram

```
O0101
N1 G90 G0 X180 Y120 F120
N2 G1 X240
N3 Y160
N4 G3 X180 Y120 R80
N5 M99
```

main program

```
O0100
N1 G90              (absolute coordinate specification)
N2 M98 P101         (call of subprogram)
N3 G51.1 X140       (mirror image applied to an axis parallel to axis Y on
                    coordinate X=140)
N4 M98 P101         (call of subprogram)
N5 G51.1 Y100       (mirror image applied to an axis parallel to axis X on
                    coordinate Y=100)
N6 M98 P101         (call of subprogram)
N7 G50.1 X0         (canceling mirror image on the axis parallel to Y)
N8 M98 P101         (call of subprogram)
N9 G50.1 Y0         (canceling mirror image on the axis parallel to X)
```

15.4 Rules of Programming Special Transformations

Rotation and scaling instructions, G68 and G51 can be issued in any order.
It should be borne in mind, however, that - when rotation is followed by scaling - the rotation command will have an effect on the coordinates of the center of scaling. If, on the other hand, scaling is followed by rotation, the scaling command will have an effect on the coordinates of the center of rotation.
Furthermore, the on and off commands of the two procedures have to be "nested", they must not overlap each other:

rotation-scaling

```
N1 G90 G17 G0 X0 Y0
N2 G68 X80 Y40 R60
N3 G51 X130 Y70 P0.5
N4 X180 Y40
N5 G1 Y100 F200
N6 X80
N7 Y40
N8 X180
N9 G50
N10 G69 G0 X0 Y0
```

scaling-rotation

```
N1 G90 G17 G0 X0 Y0
N2 G51 X130 Y70 P0.5
N3 G68 X80 Y40 R60
N4 X180 Y40
N5 G1 Y100 F200
N6 X80
N7 Y40
N8 X180
N9 G69
N10 G50 G0 X0 Y0
```

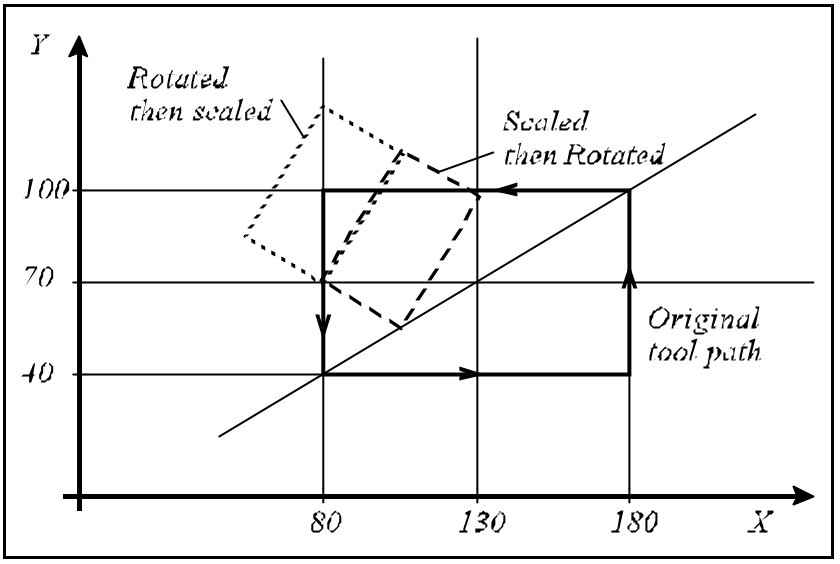


Fig. **15.4-1**

It is evident from the figure that the order of applying the various transformations is relevant.

The programmed mirror image is a different case. It can be set up in states G50 and G69 only, i.e., in the absence of scaling and rotation commands.
On the other hand, with mirror imaging set up, both scaling and rotation can be applied.
Mirror images also may not be overlapped with scaling and rotation commands. Accordingly first rotation and scaling have to be canceled in the appropriate order, followed by the instruction canceling the mirror image.

G51.1 ... (mirror image set up)
G51 ... (scaling set up)
G68 ... (rotation set up)
...

G69 ... (canceling rotation)
G50 ... (canceling scaling)
G50.1 ... (canceling mirror image)

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.

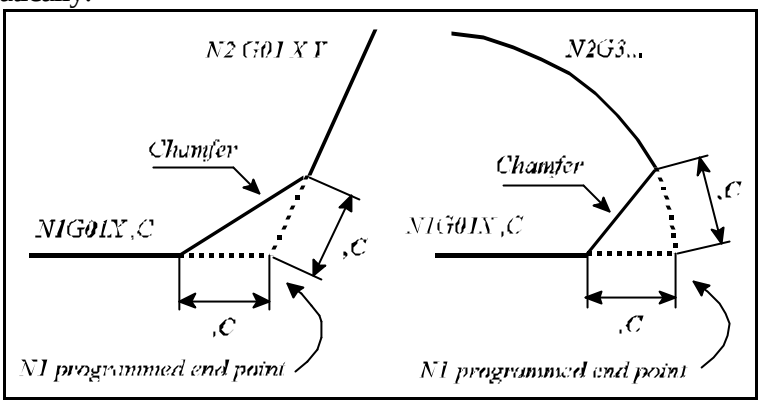


Fig. 16.1-1

E.g.:

```
N1 G1 G91 X30 ,C10
N2 X10 Y40
```

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.

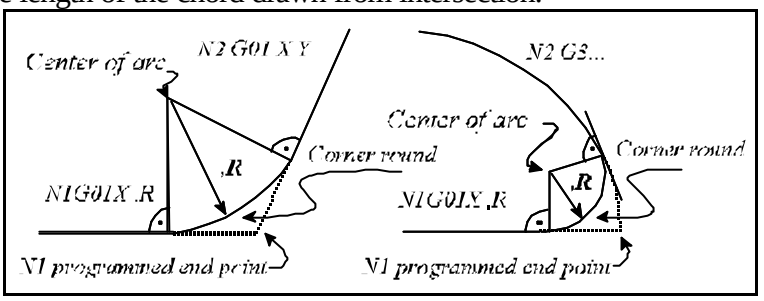


Fig. 16.1-2

E.g.:

```
N1 G91 G01 X30 ,R8
N2 G03 X-30 Y30 R30
```

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 rounding may also be written at the end of more successive blocks as shown in the below example:

```
...
G1 Y40 ,C10
X60 ,R22
G3 X20 Y80 R40 ,C10
G1 Y110
...
```

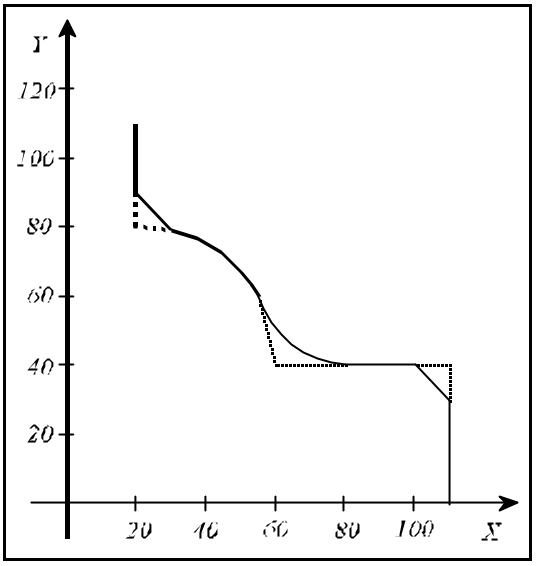


Fig. 16.1-3

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.

$$G17 \left\{\frac{G00}{G01}\right\} \left\{\frac{X_p\ ,A}{Y_p\ ,A}\right\} q\ F$$

$$G18 \left\{\frac{G00}{G01}\right\} \left\{\frac{Z_p\ ,A}{X_p\ ,A}\right\} q\ F$$

$$G19 \left\{\frac{G00}{G01}\right\} \left\{\frac{Y_p\ ,A}{Z_p\ ,A}\right\} q\ F$$

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 beside 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°.

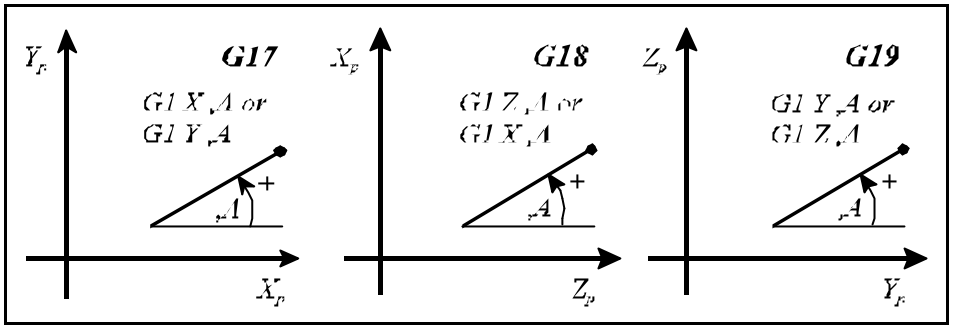


Fig. 16.2-1

For example:

```
G17 G90 G0 X57.735 Y0 ...
G1 G91...
X100 ,A30     (this specification is
               equivalent to X100 Y57.735
               where 7.735=100*tg30°)
Y100 ,A120    (this specification is
               equivalent to X-57.735 Y100
               where !57.735=100/tg120°)
X-100 ,A210   (this specification is
               equivalent to X-100 Y-57.735
               where !57.735=!100*tg30°)
Y-100 ,A300   (this specification is
               equivalent to X57.735 Y-100
               where 57.735=!100/tg120°)
```

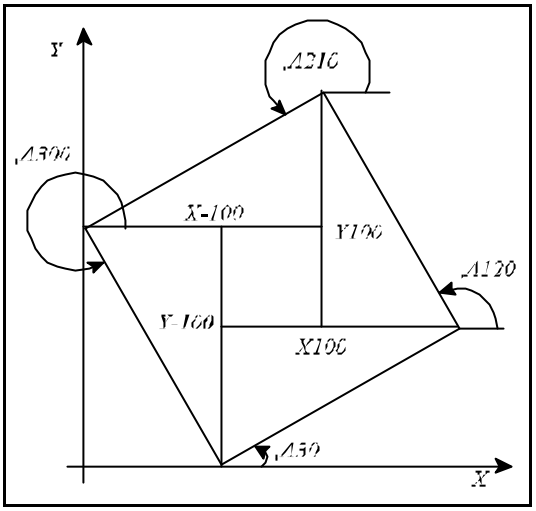


Fig. 16.2-2

L *Note:*

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

```
X100 ,A30 ,C5
Y100 ,A120 ,R10
X-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 D00 offset:

With tool radius compensation:

```
G41(or G42) ...Dnn
...
intersection calculations
...
G40
```

Without tool radius compensation:

```
G41(or G42) ...D00
...
intersection calculations
...
G40
```

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.

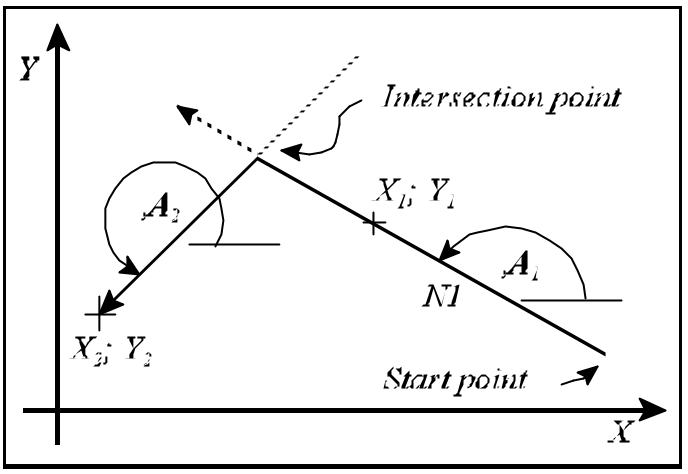


Fig. 16.3.1-1

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

```
G17 G90 G41 D0...
G0 X90 Y10
N10 G1 ,A150
N20 X10 Y20 ,A225
G0 X0 Y20
...
```

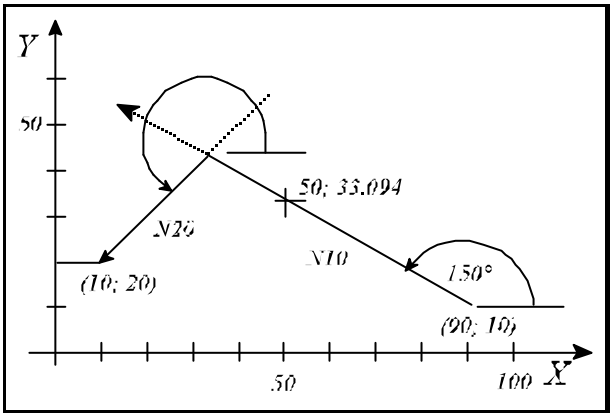


Fig. 16.3.1-2

Block N10 can also be given with the coordinates of a point of the straight line:

```
G17 G90 G41 D0...
G0 X90 Y10
N10 G1 X50 Y33.094
N20 X10 Y20 ,A225
G0 X0 Y20
...
```

Note, that in this case coordinate X, Y (X50 Y33.094) 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.:

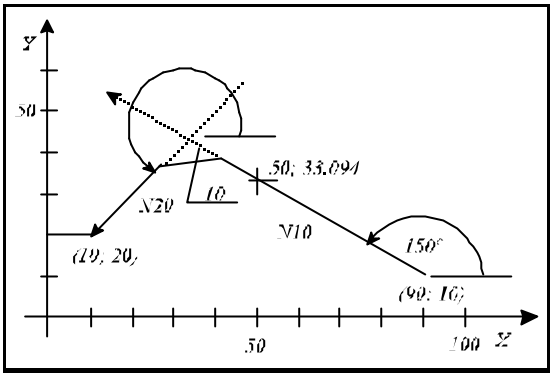


Fig. **16.3.1-3**

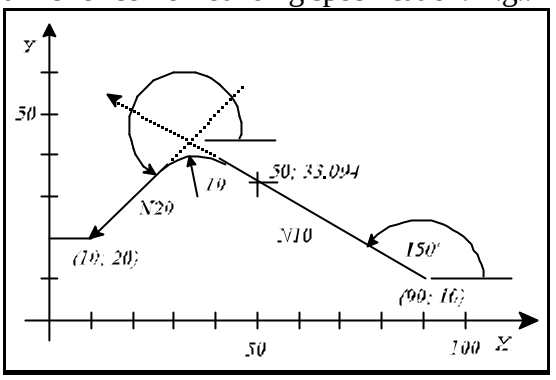


Fig. **16.3.1-4**

```
G17 G90 G41 D0...
G0 X90 Y10
N10 G1 X50 Y33.094 ,C10
N20 X10 Y20 ,A225
G0 X0 Y20
...
```

```
G17 G90 G41 D0...
G0 X90 Y10
N10 G1 X50 Y33.094 ,R10
N20 X10 Y20 ,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

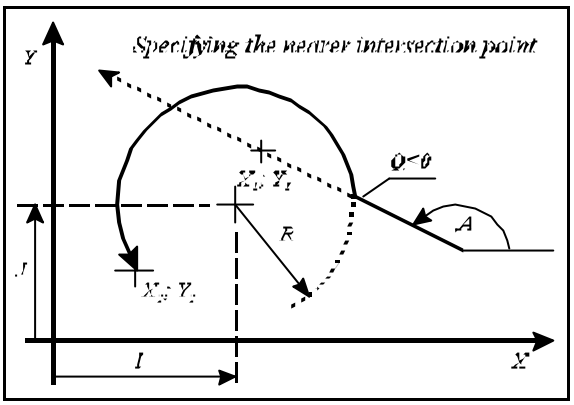


Fig. 16.3.2-1

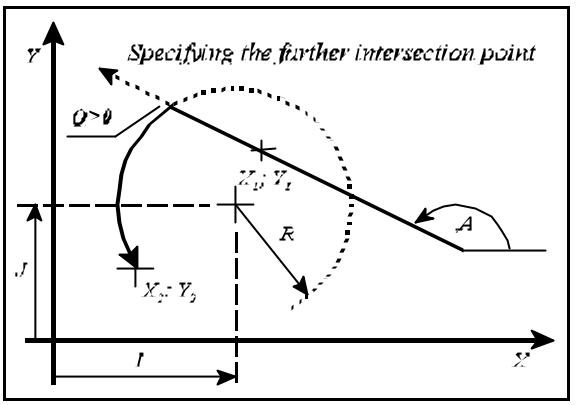


Fig. 16.3.2-2

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:

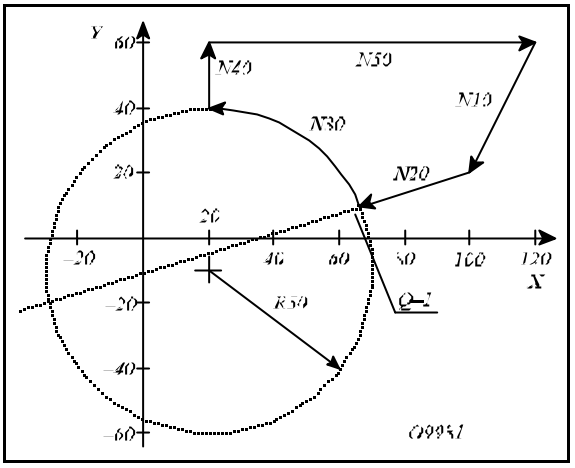


Fig. **16.3.2-3**

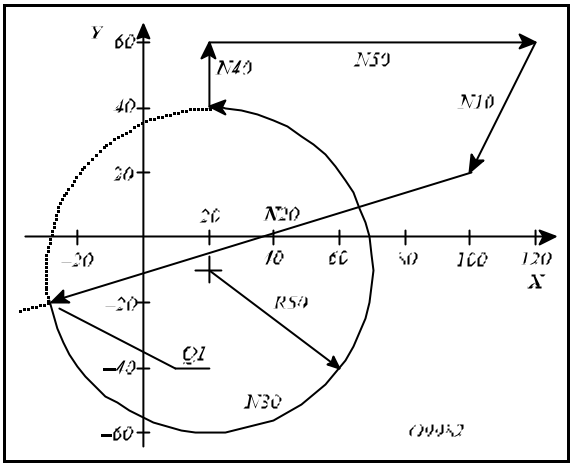


Fig. **16.3.2-4**

```
%O9981
N10 G17 G42 G0 X100 Y20 D0 S200 M3
N20 G1 X-30 Y-20
N30 G3 X20 Y40 I20 J-10 R50 Q-1
N40 G40 G0 Y60
N50 X120
N60 M30
%
```

```
%O9982
N10 G17 G42 G0 X100 Y20 D0 S200 M3
N20 G1 X-30 Y-20
N30 G3 X20 Y40 I20 J-10 R50 Q1
N40 G40 G0 Y60
N50 X120
N60 M30
%
```

Circular block N30 G3 is determined over, for both the center coordinates (I20 J–10 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 G17 G42 G0 X100 Y20 D0 S200 M3
N20 G1 X-30 Y-20 ,R15
N30 G3 X20 Y40 I20 J-10 R50 Q-1
N40 G40 G0 Y60
N50 X120
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

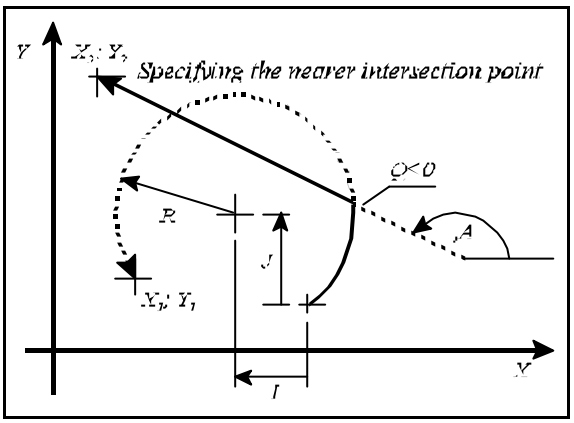


Fig. 16.3.3-1

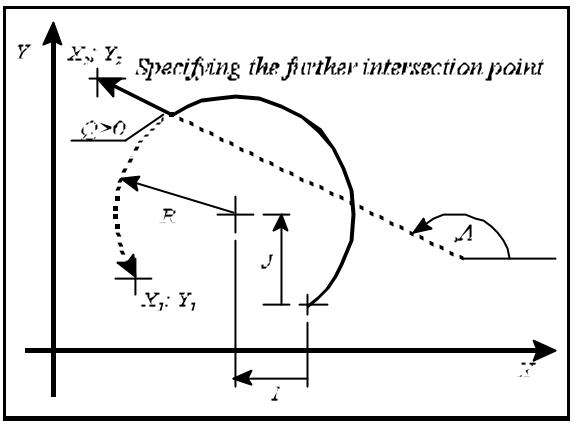


Fig. 16.3.3-2

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:

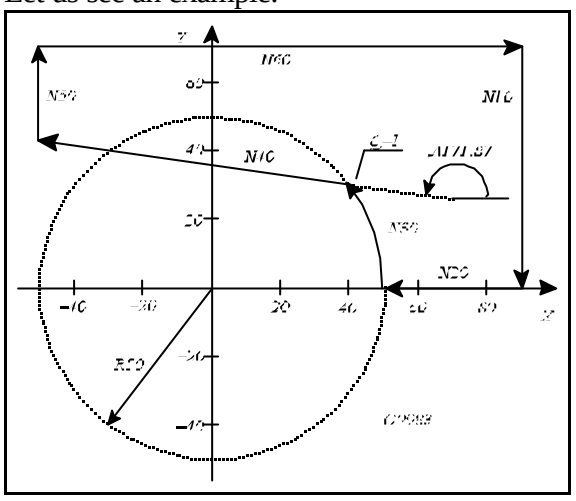


Fig. 16.3.3-3

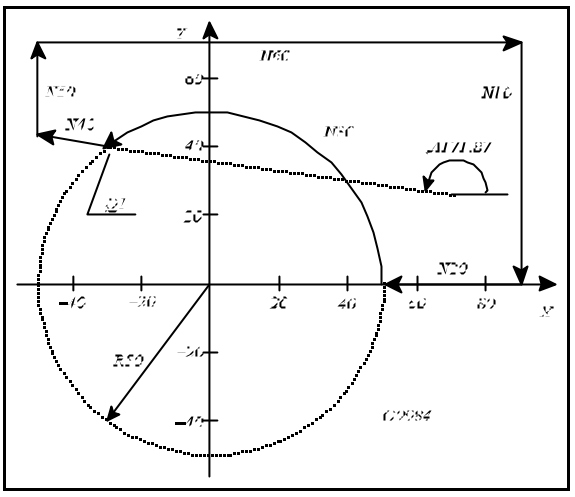


Fig. 16.3.3-4

```
%O9983
N10 G17 G0 X90 Y0 M3 S200
N20 G42 G1 X50 D0
N30 G3 X-50 Y0 R50
N40 G1 X-50 Y42.857 ,A171.87 Q-1
N50 G40 G0 Y70
N60 X90
N70 M30
%
```

```
%O9984
N10 G17 G0 X90 Y0 M3 S200
N20 G42 G1 X50 D0
N30 G3 X-50 Y0 R50
N40 G1 X-50 Y42.857 ,A171.87 Q1
N50 G40 G0 Y70
N60 X90
N70 M30
%
```

Linear block N40 is defined over because both the end point coordinates (X–50 Y42.875) and the angle (,A171.87) of the straight line are specified. Therefore X–50 Y0 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 G17 G0 X90 Y0 M3 S200
N20 G42 G1 X50 D0
N30 G3 X-50 Y0 R50 ,R15
N40 G1 X-50 Y42.857 ,A171.87 Q-1
N50 G40 G0 Y70
N60 X90
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

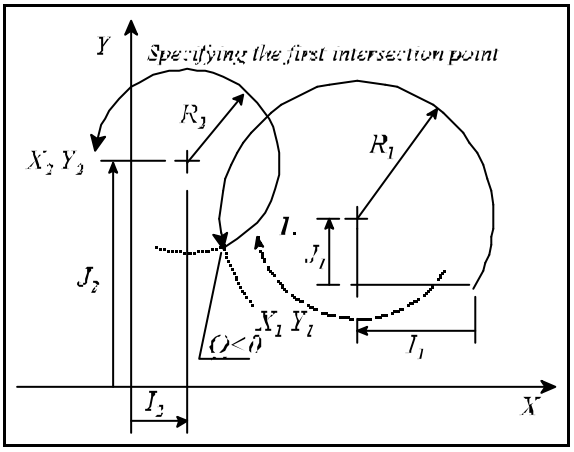


Fig. 16.3.4-1

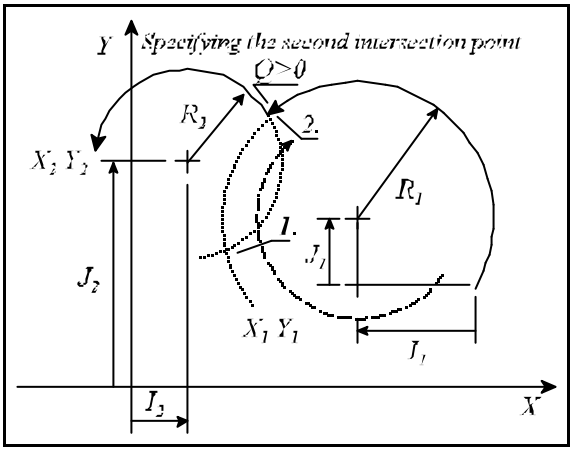


Fig. 16.3.4-2

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:

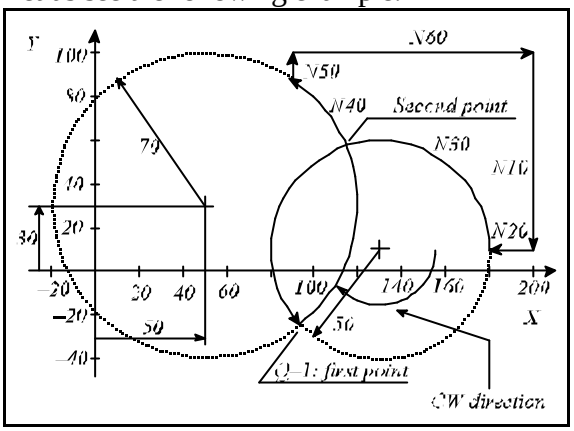


Fig. 16.3.4-3

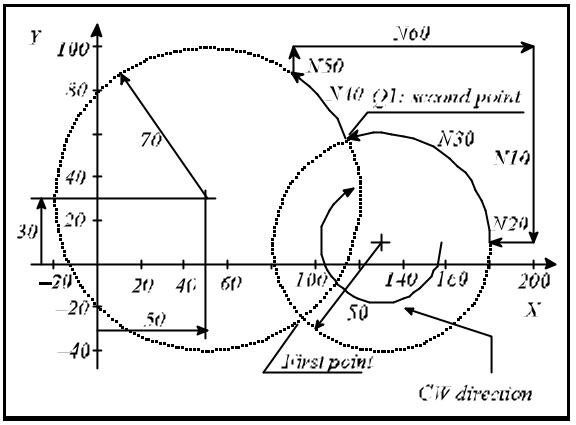


Fig. 16.3.4-4

```
%O9985
N10 G17 G54 G0 X200 Y10 M3 S200
N20 G42 G1 X180 D1
N30 G3 X130 Y-40 R-50
N40 X90 Y87.446 I50 J30 R70 Q–1
N50 G40 G0 Y100
N60 X200
N70 M30
%
```

```
%O9986
N10 G17 G54 G0 X200 Y10 M3 S200
N20 G42 G1 X180 D1
N30 G3 X130 Y-40 R-50
N40 X90 Y87.446 I50 J30 R70 Q1
N50 G40 G0 Y100
N60 X200
N70 M30
%
```

Circular block N40 is defined over because both, the center coordinates (I50 J30 in absolute value) and the radius (R70) of the circle, are specified. Therefore coordinates X130 Y–40 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 G17 G54 G0 X200 Y10 M3 S200
N20 G42 G1 X180 D1
N30 G3 X130 Y-40 R-50 ,R20
N40 X90 Y87.446 I50 J30 R70 Q1
N50 G40 G0 Y100
N60 X200
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:

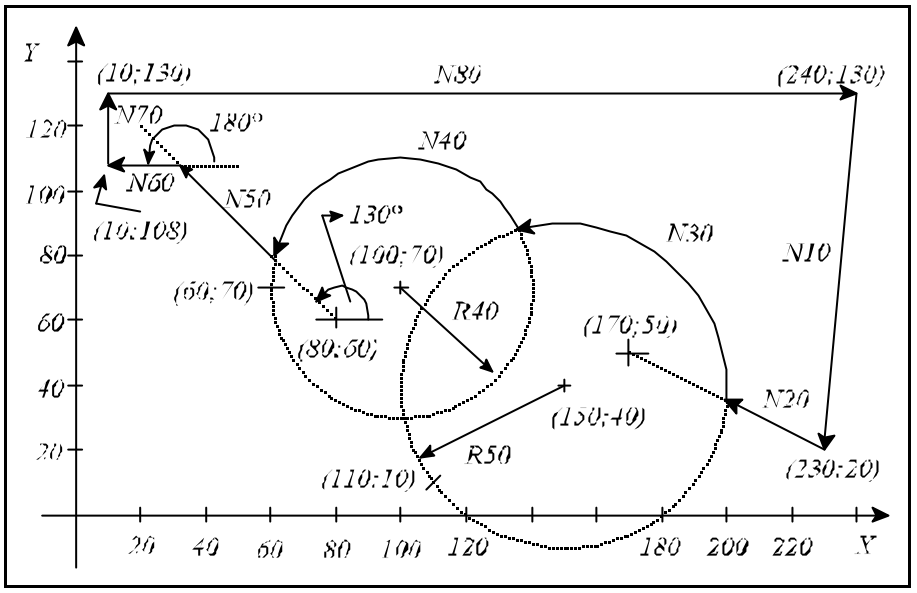


Fig. **16.3.5-1**

```
%O9984
N10 G17 G54 G0 G42 X230 Y20 D1 F300 S500 M3
N20 G1 X170 Y50
N30 G3 X110 Y10 I150 J40 R50 Q-1
N40 X60 Y70 I100 J70 R40 Q1
N50 G1 X80 Y60 ,A135 Q1
N60 X10 Y108 ,A180
N70 G40 G0 Y130
N80 X240
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 (X170 Y50) 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 (X110 Y10) 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 X10 Y108 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 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.

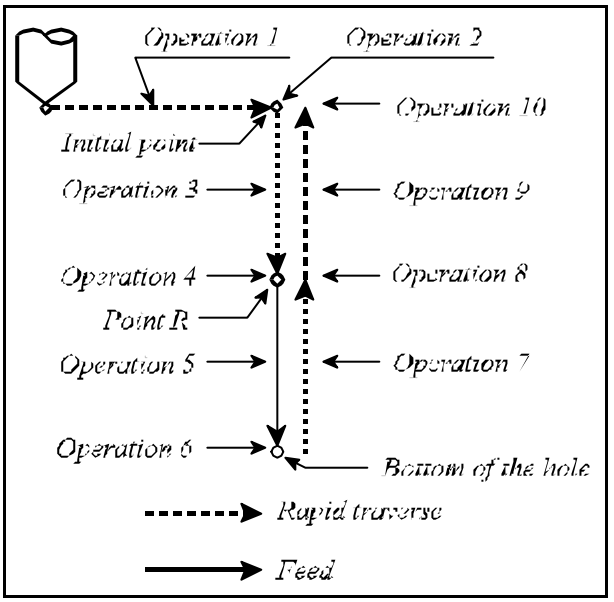


Fig. 17-1

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.

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.

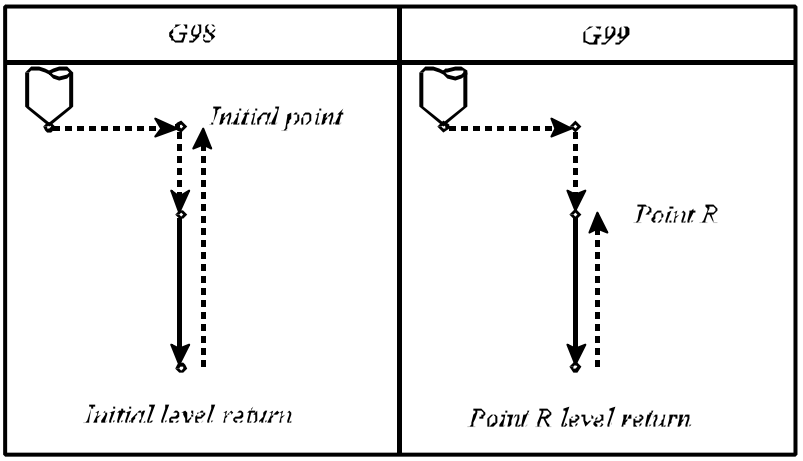


Fig. 17-2

Codes of the drilling cycles: G73, G74, G76, 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_      I_ J_   Zp_ R_ Q_ E_ P_ F_ S_     L_
G18  G_   Zp_ Xp_      K_ I_   Yp_ R_ Q_ E_ P_ F_ S_     L_
G19  G_   Yp_ Zp_      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 G73, G74, G76, 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 Y0 Z50 R150
N3 X100 Y30 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 Y0 Z50 R150
N3 X100 Y30 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

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.

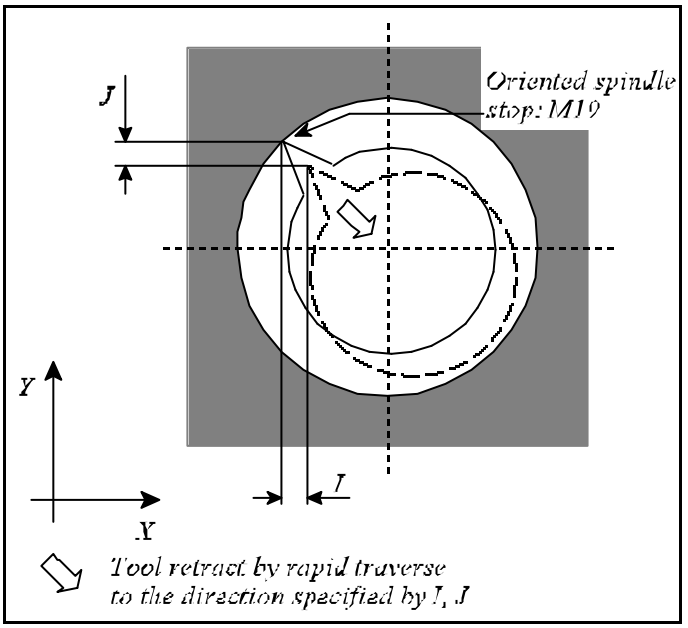


Fig. 17-3

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 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.

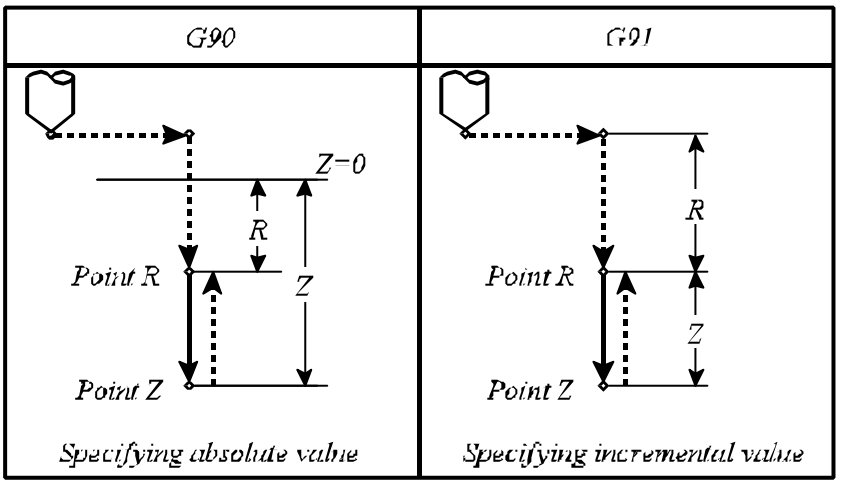


Fig. 17-4

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_ Y_ 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 Y_ Z_ P_
```

The position of the hole is shifted in direction Y. 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 Y_ 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_ Y_
```

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 G17 G0 X0 Y0 Z100 M3
N2 G91 G81 X100 Z-40 R-97 F50 L5
```

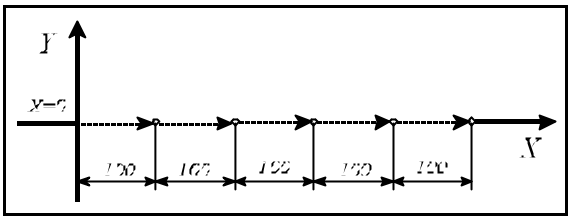


Fig. 17-5

Under the above instructions the control will drill 5 identical holes spaced at 100mm along axis X. The position of the first hole is X=100, Y=0.

Under G91 the position of the hole has been specified incrementally. If it had been specified as an absolute data (G90), the operation would have been carried out five times in succession in the point of X100, Y0 coordinates.

```
N1 G90 G17 G16 G0 X200 Y-60 Z50 M3
N2 G81 YI60 Z-40 R3 F50 L6
```

Under the above instructions the control will drill 6 holes spaced at 60 degrees around a circle of a 200mm radius. The position of the first hole coincides with the point of X=200 Y=0 coordinates.

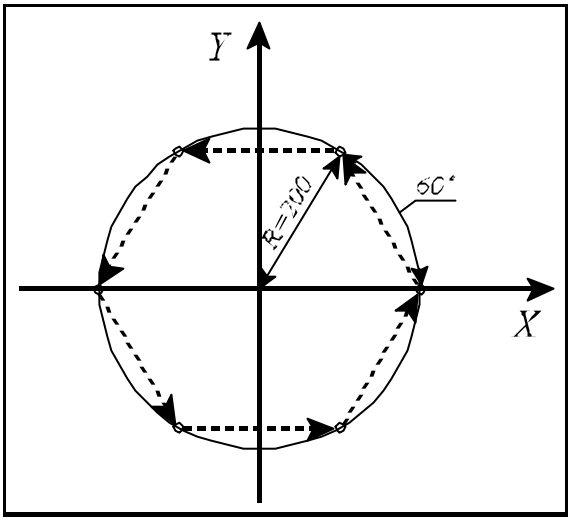


Fig. 17-6

17.1 Detailed Description of Canned Cycles

17.1.1 High Speed Peck Drilling Cycle (G73)

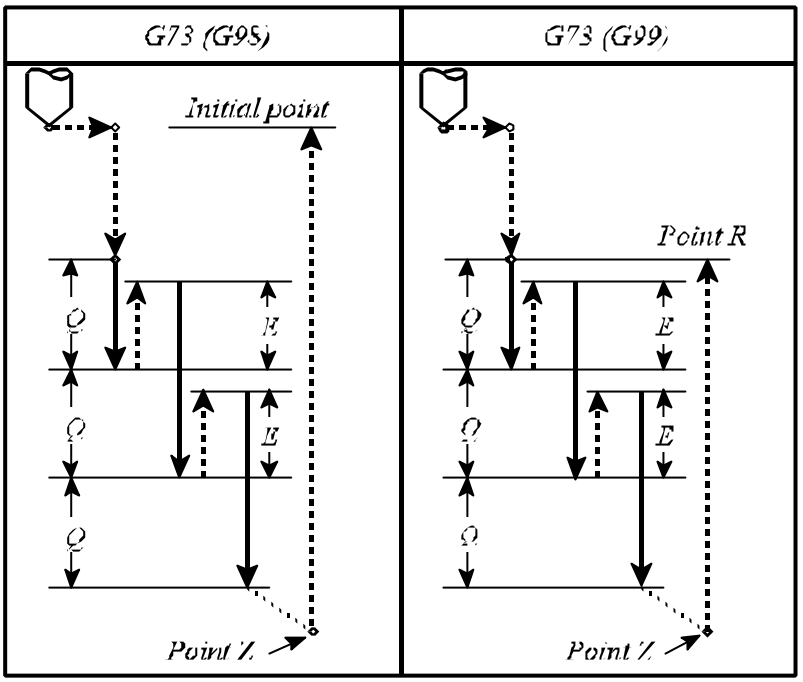


Fig. 17.1.1-1

The variables used in the cycle are

G17 **G73** X_p__ Y_p__ Z_p__ R__ Q__ E__ F__ L__
G18 **G73** Z_p__ X_p__ Y_p__ R__ Q__ E__ F__ L__
G19 **G73** Y_p__ Z_p__ 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.

17.1.2 Counter Tapping Cycle (G74)

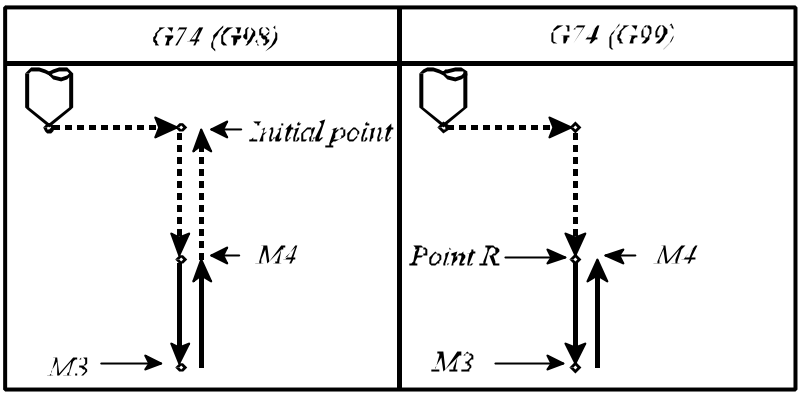


Fig. 17.1.2-1

This cycle can be used only with a spring tap. The variables used in the cycle are

G17 **G74** X_p__ Y_p__ Z_p__ R__ (P__) F__ L__
G18 **G74** Z_p__ X_p__ Y_p__ R__ (P__) F__ L__
G19 **G74** Y_p__ Z_p__ 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. -

17.1.3 Fine Boring Cycle (G76)

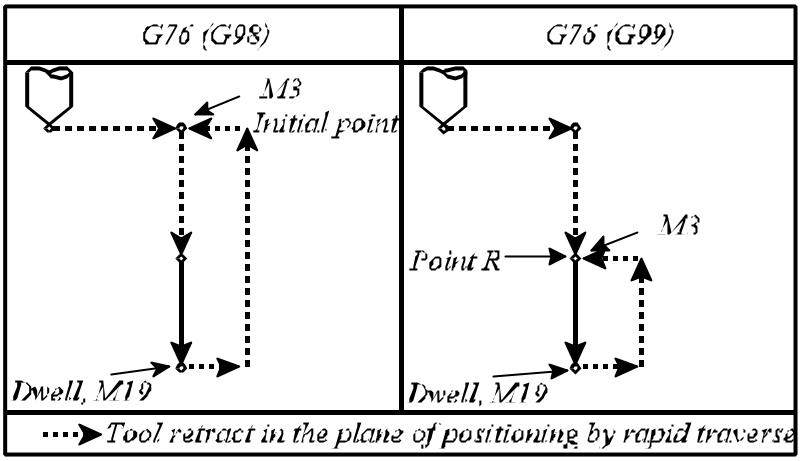


Fig. 17.1.3-1

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 **G76** X_p__ Y_p__ I__ J__ Z_p__ R__ P__ F__ L__
G18 **G76** Z_p__ X_p__ K__ I__ Y_p__ R__ P__ F__ L__
G19 **G76** Y_p__ Z_p__ 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

17.1.4 Canned Cycle 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).

17.1.5 Drilling, Spot Boring Cycle (G81)

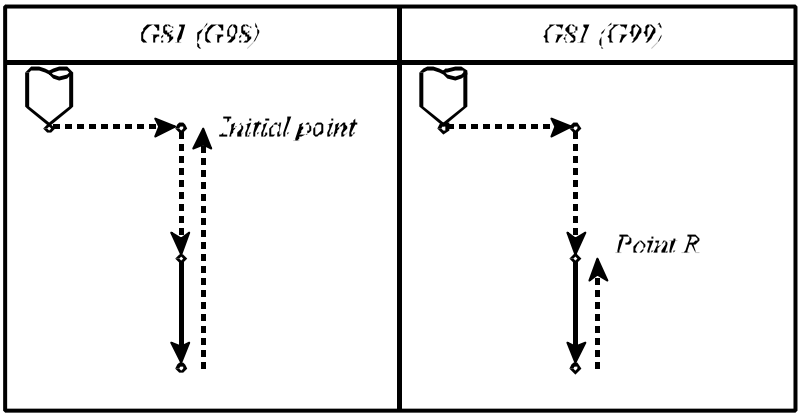


Fig. 17.1.5-1

The variables used in the cycle are

G17 **G81** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G81** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G81** Y_p__ Z_p__ X_p__ R__ F__ L__

The operations of the cycle are

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. -

17.1.6 Drilling, Counter Boring Cycle (G82)

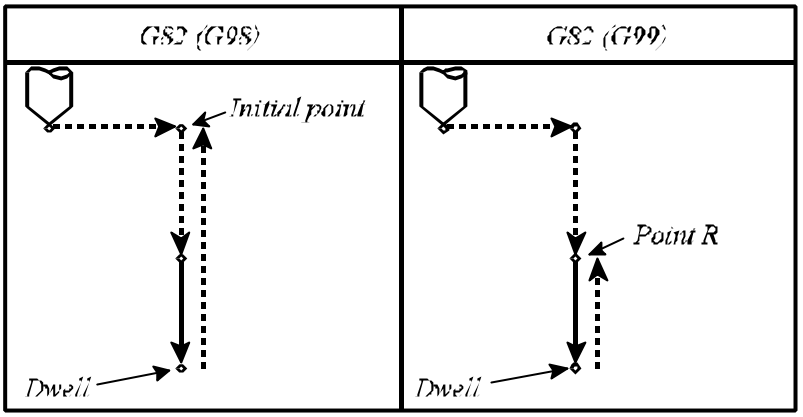


Fig. 17.1.6-1

The variables used in the cycle are

G17 **G82** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G82** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G82** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

the operations of the cycle are

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. -

17.1.7 Peck Drilling Cycle (G83)

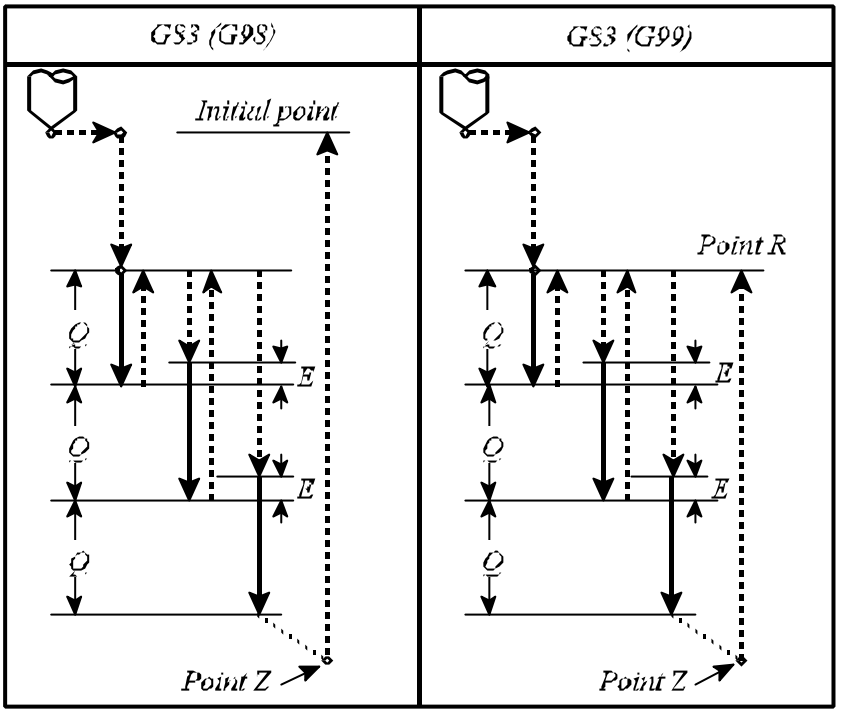


Fig. 17.1.7-1

The variables used in the cycle are

G17 **G83** X_p__ Y_p__ Z_p__ R__ Q__ E__ F__ L__
G18 **G83** Z_p__ X_p__ Y_p__ R__ Q__ E__ F__ L__
G19 **G83** Y_p__ Z_p__ X_p__ R__ Q__ E__ F__ L__

The 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*.

17.1.8 Tapping Cycle (G84)

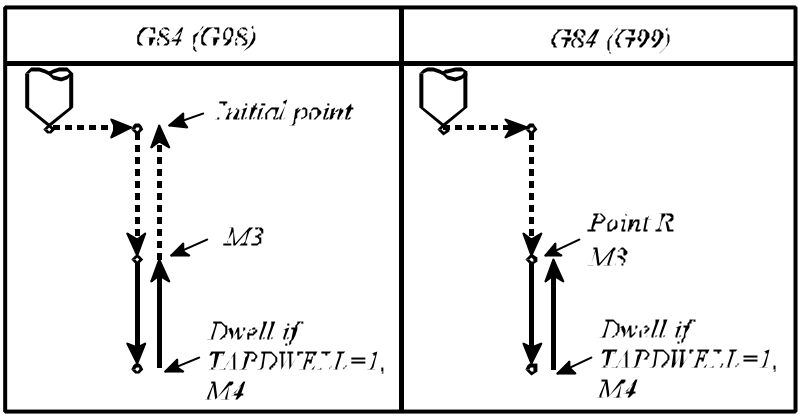


Fig. 17.1.8-1

This cycle can be used only with a spring tap.
The variables used in the cycle are

G17 **G84** X_p__ Y_p__ Z_p__ R__ (P__) F__ L__
G18 **G84** Z_p__ X_p__ Y_p__ R__ (P__) F__ L__
G19 **G84** Y_p__ Z_p__ X_p__ R__ (P__) F__ L__

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):

$$F = P * S$$

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. -

17.1.9 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__ Z_p__ R__ F__ S__ L__
G18 **G84._** Z_p__ X_p__ Y_p__ R__ F__ S__ L__
G19 **G84._** Y_p__ Z_p__ 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)	$F_s=S \cdot 360 \left(\frac{degrees}{min} \right)$

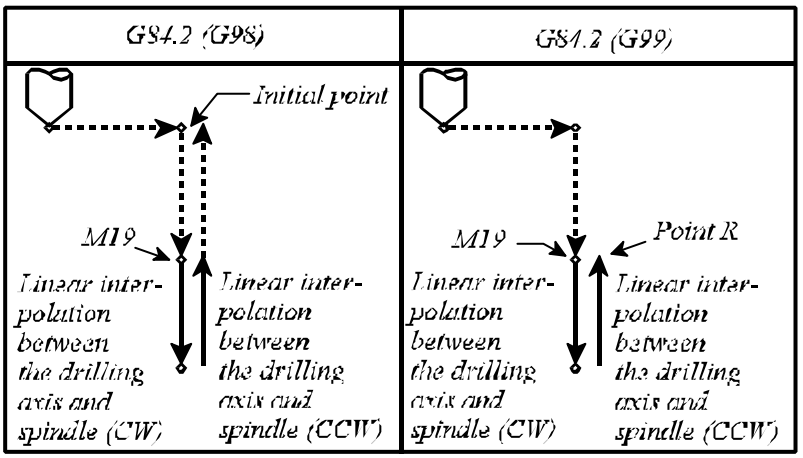


Fig. 17.1.9-1

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. -

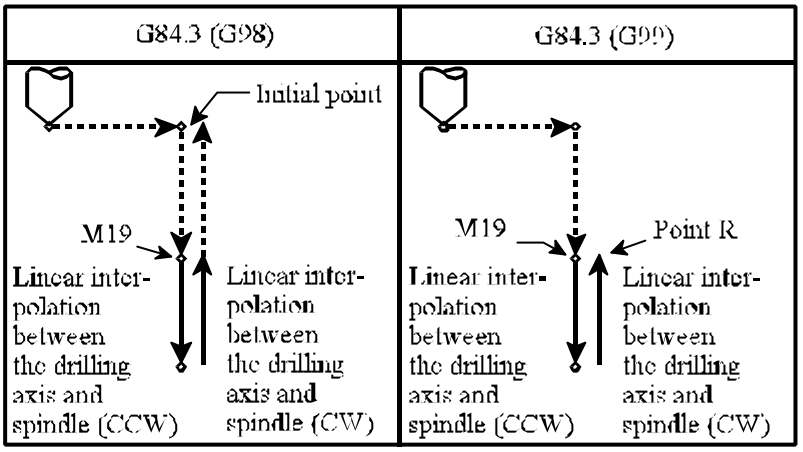


Fig. 17.1.9-2

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. -

17.1.10 Boring Cycle (G85)

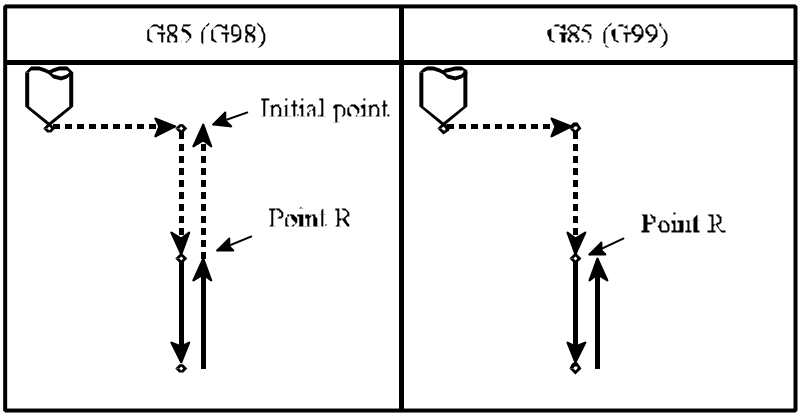


Fig. **17.1.10-1**

The variables used in the cycle are

G17 **G85** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G85** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G85** Y_p__ Z_p__ 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. -

17.1.11 Boring Cycle Tool Retraction with Rapid Traverse (G86)

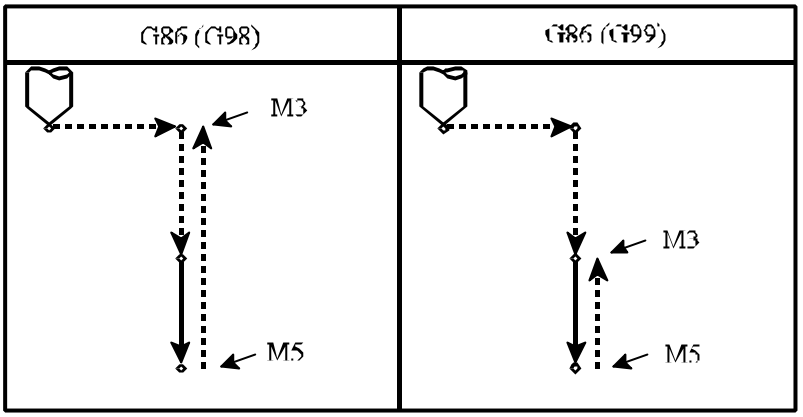


Fig. **17.1.11-1**

The variables used in the cycle are

G17 **G86** X_p__ Y_p__ Z_p__ R__ F__ L__
G18 **G86** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G86** Y_p__ Z_p__ X_p__ R__ F__ L__

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)

17.1.12 Boring Cycle/Back Boring Cycle (G87)

The cycle will be performed in two different ways.

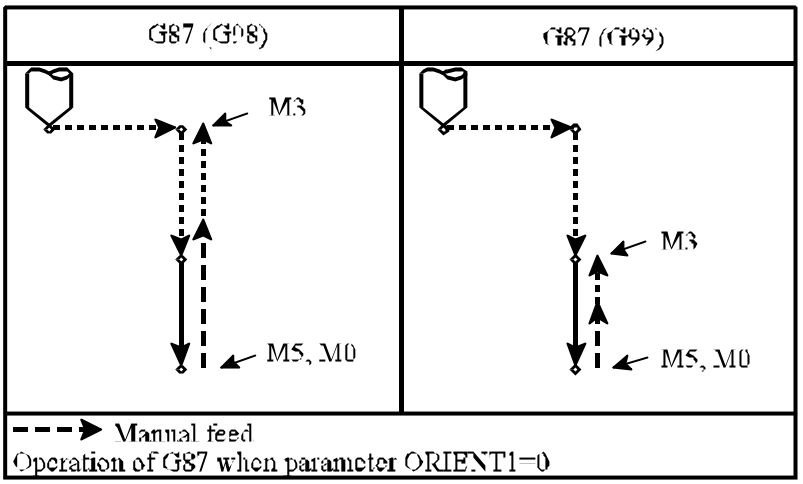


Fig. 17.1.12-1

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__ Z_p__ R__ F__ L__
G18 **G87** Z_p__ X_p__ Y_p__ R__ F__ L__
G19 **G87** Y_p__ Z_p__ X_p__ R__ F__ L__

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)

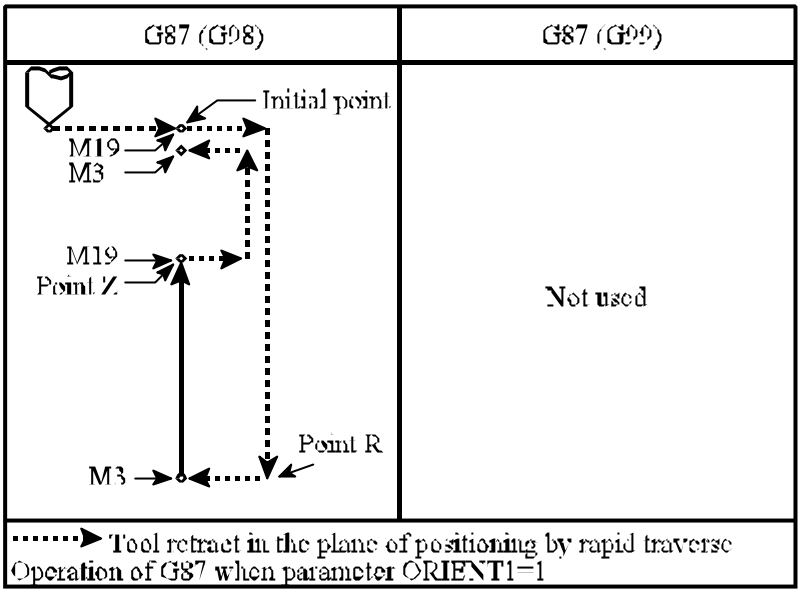


Fig. 17.1.12-2

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__ I__ J__ Z_p__ R__ F__ L__
G18 **G87** Z_p__ X_p__ K__ I__ Y_p__ R__ F__ L__
G19 **G87** Y_p__ Z_p__ J__ K__ X_p__ R__ F__ L__

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.

17.1.13 Boring Cycle (Manual Operation on the Bottom Point) (G88)

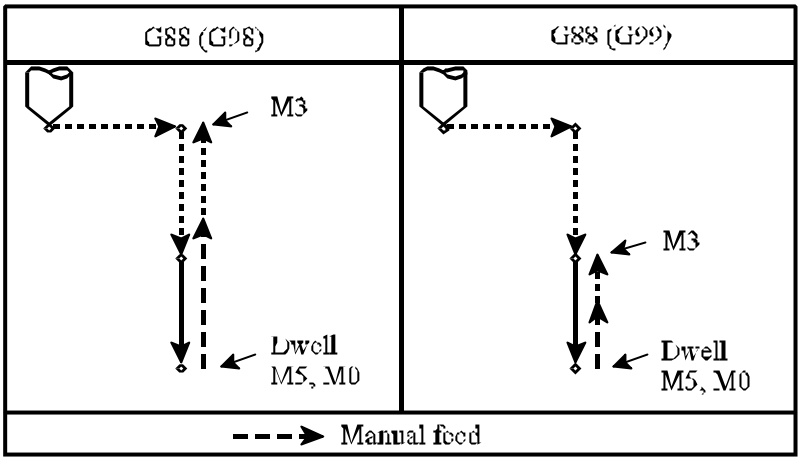


Fig. 17.1.13-1

The variables used in the cycle are

G17 **G88** X_p__ Y_p__ Z_p__ R__ P__ F__ L__
G18 **G88** Z_p__ X_p__ Y_p__ R__ P__ F__ L__
G19 **G88** Y_p__ Z_p__ X_p__ R__ P__ F__ L__

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.

17.1.14 Boring Cycle (Dwell on the Bottom Point, Retraction with Feed) (G89)

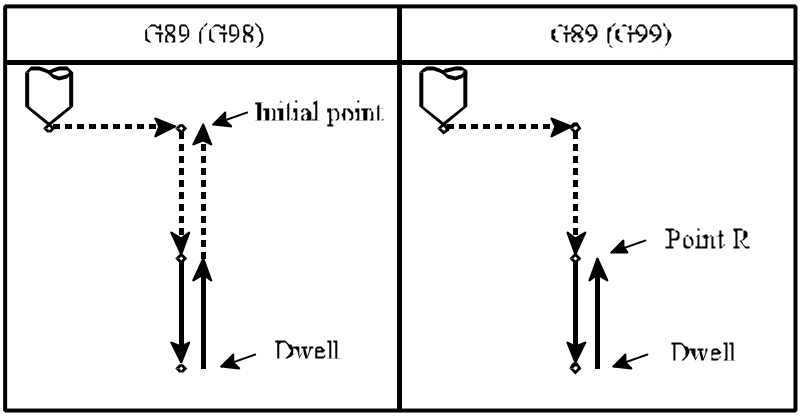


Fig. 17.1.14-1

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.

17.2 Notes to the use of canned cycles

– 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, 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_ Y_ 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**Q**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 G74, 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 G43, G44, G49 is programmed in a cycle interpolation, or if a new value of H is specified, the length compensation will be taken into account in operation 3, invariably along the drilling axis.
– Instructions G45, ... G48 will not be executed in the drilling cycle.

18 Measurement Functions

18.1 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

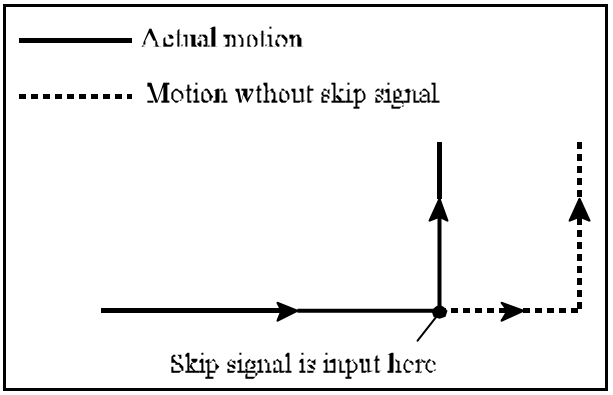


Fig. **18.1-1**

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),
– to be understood invariably in the current work coordinate system,
– with the actual length compensation (G43, G44) and
– with the actual tool offset (G45 ... G48) taken into account.

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 X100
N2 X30 Y50
```

An incremental motion in direction X is started in block N1. If the control comes to a halt at the point of coordinate X=86.7 on arrival of the external signal, it will move incrementally 30 in X direction and 50 in Y direction in block N2 (reckoned from that point).

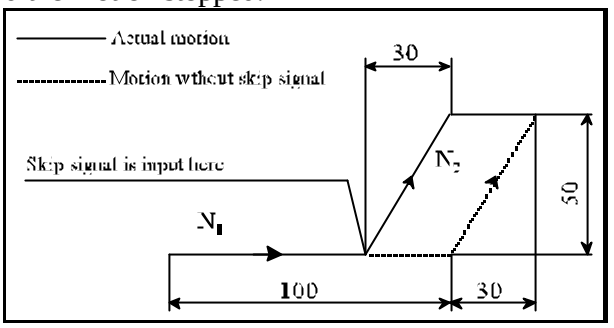


Fig. 18.1-2

In the case of an absolute data specification being programmed, the motion will be

```
N1 G31 G90 X200
N2 X300 Y100
```

Interpolation N1 starts a motion in direction X to the point of coordinate X=200. If, after arrival of the external signal, the control comes to a halt at the point of coordinate X=167, the displacement in direction X will be X=300-167, i.e., X=133 in block N2.

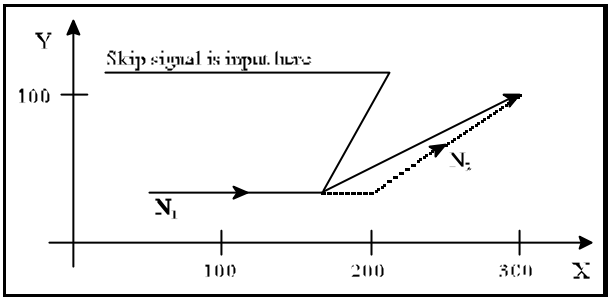


Fig. 18.1-3

18.2 Automatic Tool Length Measurement (G37)

Instruction

G37 q

will cause the motion to be started in rapid traverse in the direction specified at coordinate q. The value of q 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 q - *RAPDIST* where *RAPDIST* is a parameter-selected value.
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 arrives outside of the ALADIST range (specified on parameter) of the predicted position q.
If the measurement is completed successfully

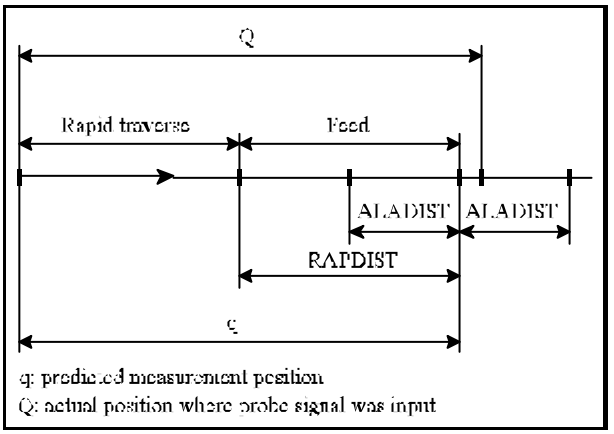


Fig. 18.2-1

and the touch-probe signal has arrived at the point of coordinate Q, the control will
– add the difference Q-q to the wear of compensation register selected on address H earlier (if parameter *ADD*=1)
– or will subtract the difference from it (if parameter *ADD*=0).

The appropriate H value and the length compensation have to be set up prior to commencement of the measurement.
– G37 is a single-shot instruction.
– Cycle 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 G referring to a length compensation (G43, G44, G49) 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 or G16 is in effect.

The following error message will be returned during the execution of function G37.
– Message *3103 OUT OF RANGE* is returned if the touch-probe signal arrives outside of the *ALADIST* range of the end position programmed in interpolation G37.

19 Safety Functions

19.1 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:

XI, YJ, Z$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.

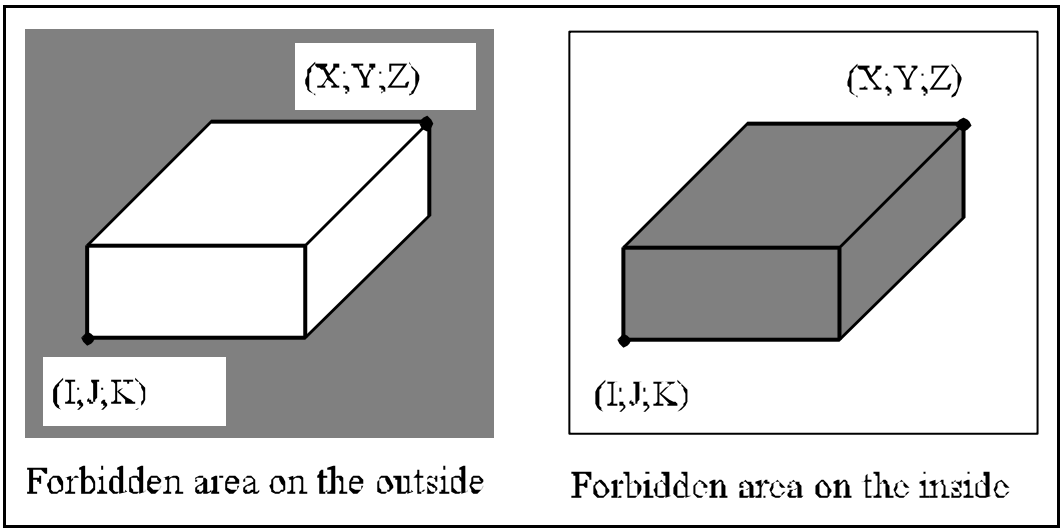


Fig. **19.1-1**

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.

19.2 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.

S Parametric overtravel function is only performed by the control after reference point has been returned.
S The parametric overtravel function will prohibit always an external area.
S The areas of programmed stroke check and that of overtravel functions may overlap.

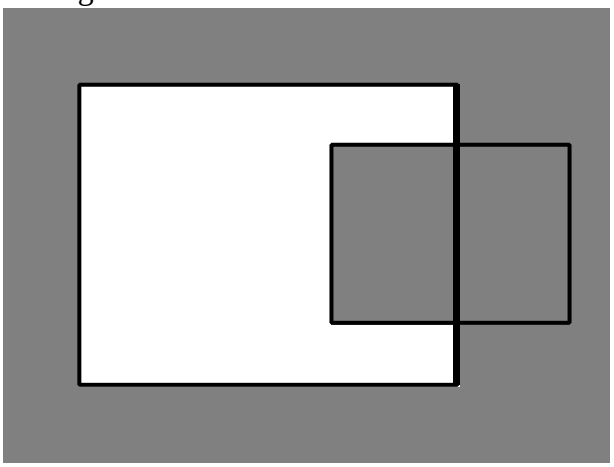


Fig. 19.2-1

19.3 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.

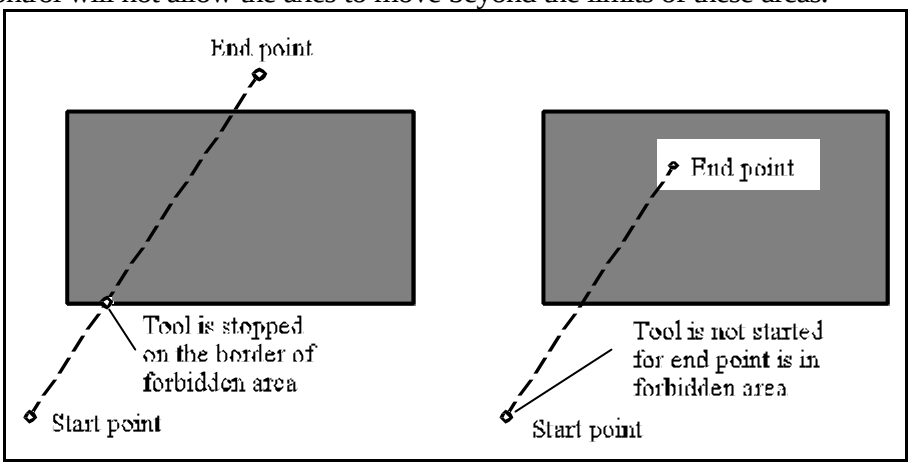


Fig. 19.3-1

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.

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.

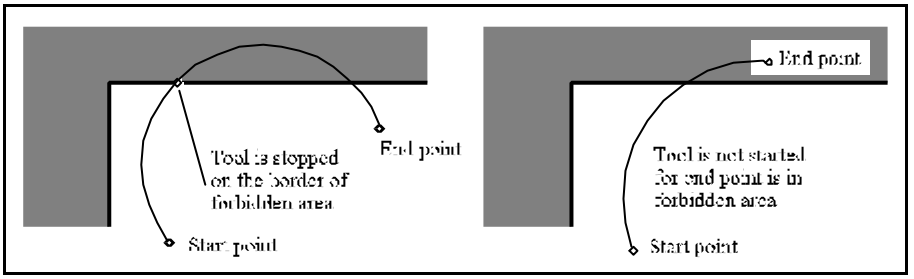


Fig. 19.3-2

20 Custom Macro

20.1 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

l v	*1. a a*	*2. a a*
#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

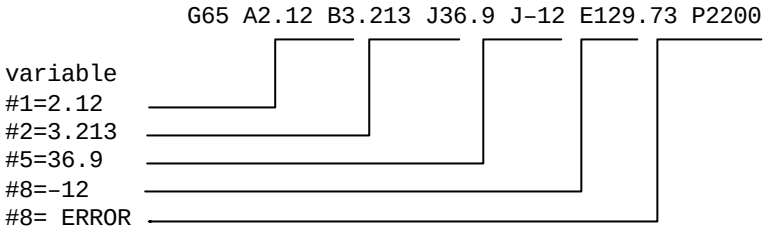
l v	*1. a a*	*2. a a*
#23	W	J7
#24	X	K7
#25	Y	I8
#26	Z	J8
#27	–	K8
#28	–	I9
#29	–	J9
#30	–	K9
#31	–	I10
#32	–	J10
#33	–	K10

S **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,



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.

20.2 The Modal Macro Call

20.2.1 Macro Modal Call in Every Motion Command (G66)

As a result of instruction

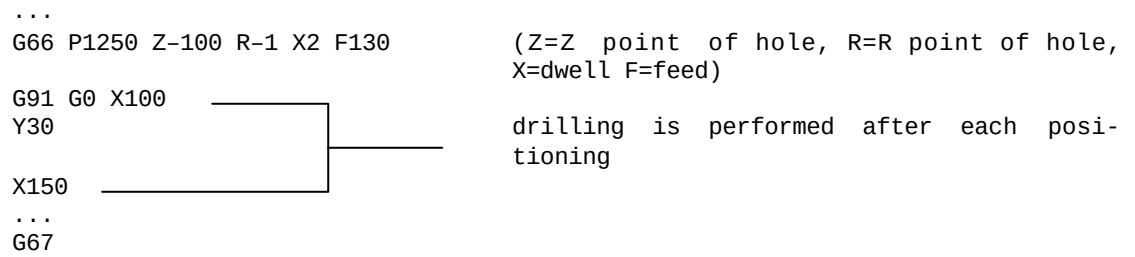
G66 P(program number) L(number of repetitions) <argument assignment>

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



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)
%
```

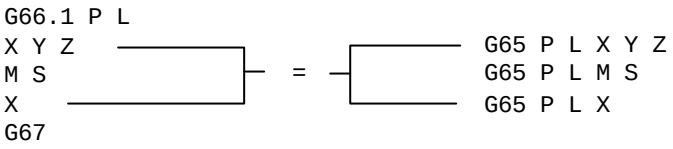
20.2.2 Macro Modal Call From Each Block (G66.1)

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:



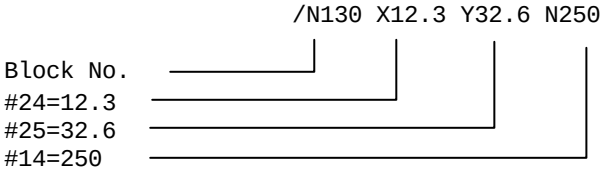
The selected macro will be called until with command

G67

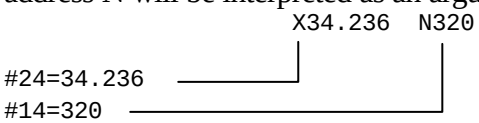
it is canceled.

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:



   if address N is in the middle of the block (preceded by any address other than "/"), address N will be interpreted as an argument:



   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.

20.3 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.

20.4 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.

20.5 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.

20.6 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.

20.7 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.

20.8 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.

20.9 Differences Between the Call of a Sub-Program 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.

20.9.1 Multiple Calls

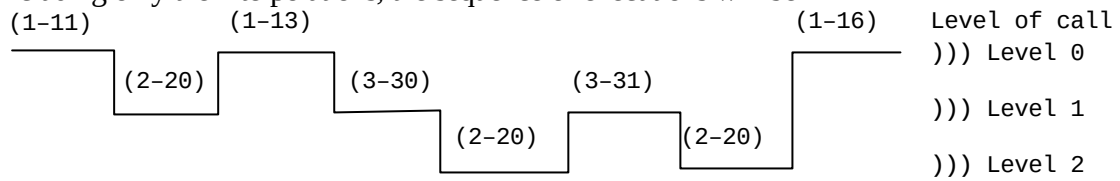
Another macro can be called again from a macro. Macro calls can be made in four levels of depth, including simple and modal ones. With the subprogram calls included, the maximum depth of the calls may cover 8 levels.
In the case of multiple calls of modal macros (type G66), first the latter specified 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 executions 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.

20.10 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.

20.11 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.

20.11.1 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.

20.11.2 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

20.11.3 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>

20.11.4 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
```

20.12 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.

20.12.1 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.

20.12.2 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.

20.12.3 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 #13999, or values can be assigned them.

No. of compensation	H		D	
	geometry	*wear*	*geometry*	*wear*
1	#10001	#11001	#12001	#13001
2	#10002	#11002	#12002	#13002
:	:	:	:	:
999	#10999	#11999	#12999	#13999

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	
:		
#5206	common work zero point offset, axis 6	
#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

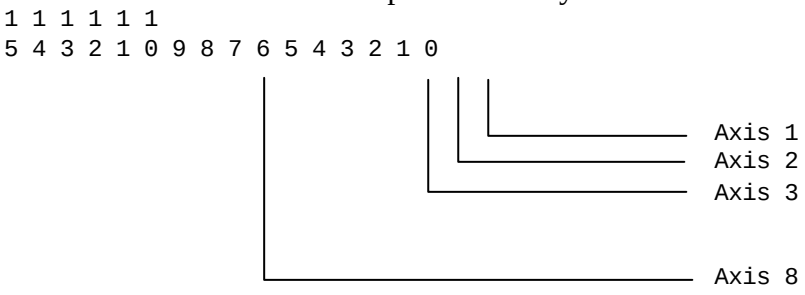
0 = function is effective
1 = function is suppressed

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.



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.

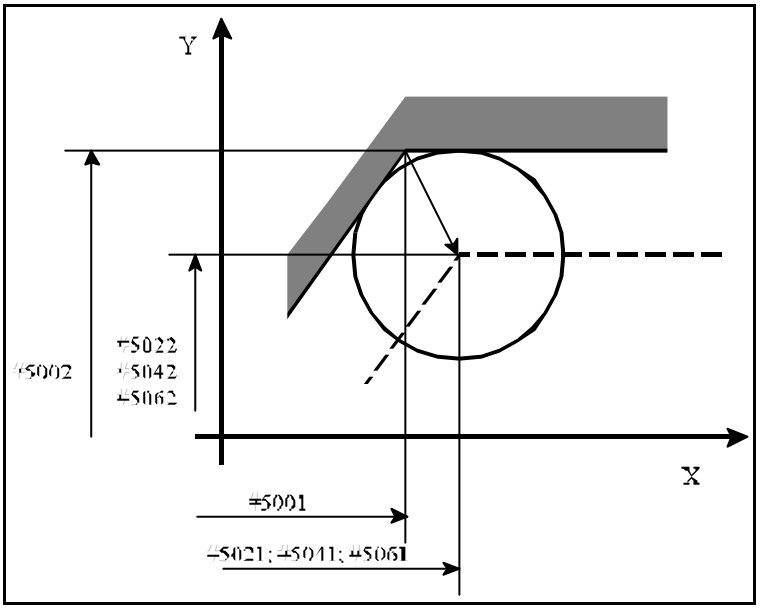


Fig. 20.12.3-1

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.

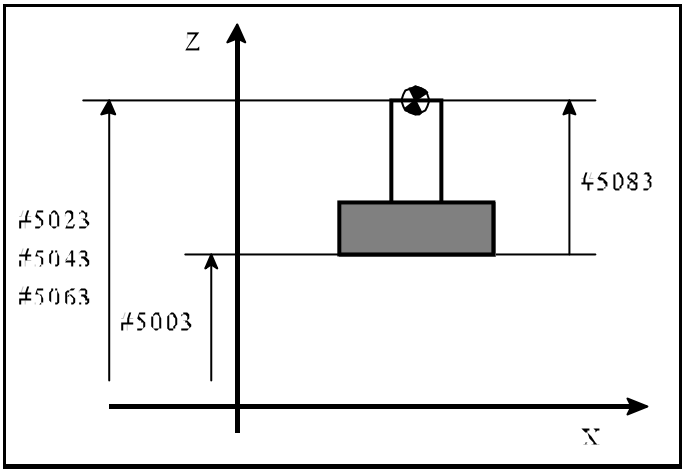


Fig. 20.12.3-2

Servo lag

system variable	nature of position information	entry during 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.

20.13 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.

20.13.1 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.

20.13.2 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##j#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##j#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.
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.

20.13.3 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.

L *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.

20.13.4 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.

20.13.5 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.

20.13.6 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.

20.13.7 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
:
```

20.13.8 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] S))) Q 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.

L *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.

20.14 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).

20.15 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.

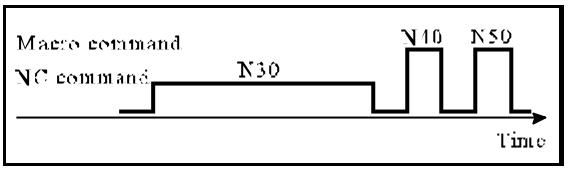


Fig. 20.15-1

L 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.

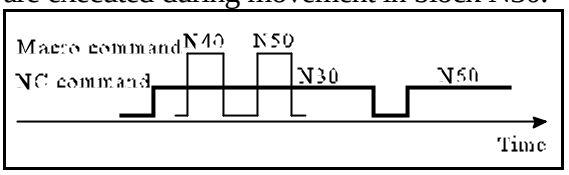


Fig. 20.15-2

L 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.

20.16 Displaying Macros and Sub-programs 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.

20.17 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.

20.18 Pocket-milling Macro Cycle

Instruction

G65 P9999 X Y Z I J K R F D E Q M S T

will start a pocket-milling cycle. For the execution of the cycle, macro of program number O9999 has to be filled in the memory, from the PROM memory of the control.

Prior to calling the cycle, the tool must be brought over the geometric center of the pocket in the selected plane, at a safety distance over the workpiece. At the end of the cycle the tool will be retracted to the same point.

The addresses in the block have the following meaning:

X = size of pocket in direction X

Y = size of pocket in direction Y

Z = size of pocket in direction Z

Instructions G17, G18, G19 will define the length, width and depth of the pocket for the three coordinates. For example, in case of G17 Z will be the depth of the pocket, the longer one of X and Y will be the length of pocket (the shorter one will be the width thereof). Those values have to be entered in absolute values as positive numbers.

R = the radius of the corners of the pocket.

Rounding (if any) of the corners of the pocket should be specified at address R. Unless address R is filled, the rounding of the pocket's corners will be rounded with the tool radius.

I = safety distance toward the depth of pocket in the case of G19

J = safety distance toward the depth of pocket in the case of G18

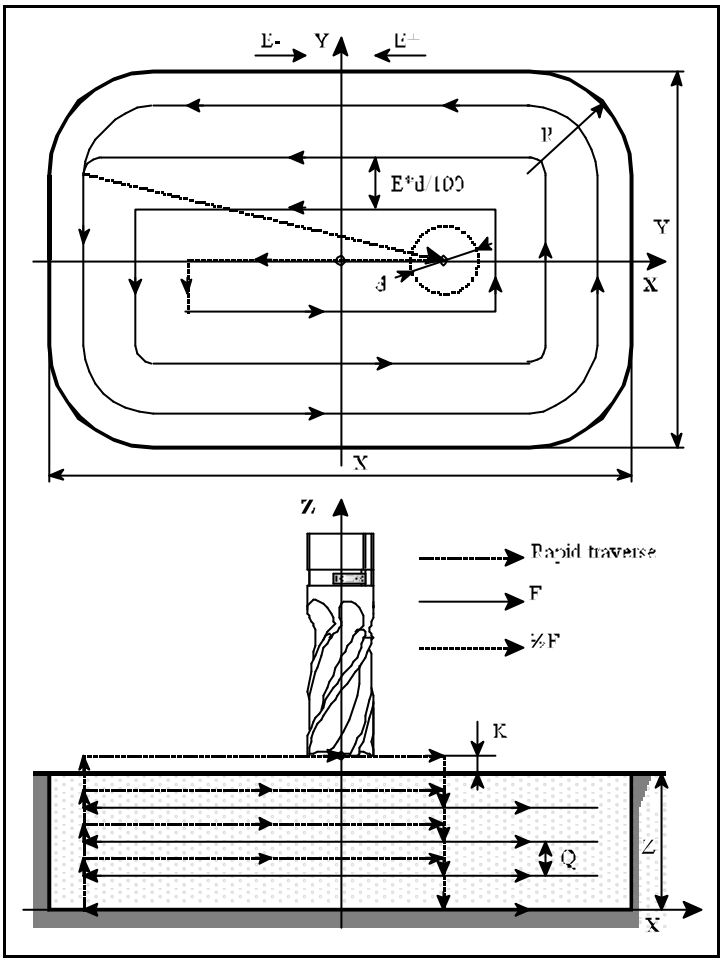


Fig. 20.18-1

K = safety distance toward the depth of pocket in the case of G17

Depending on the plane selected, the safety allowance in the direction of the tool has to be specified at the addresses I(G19), J(G18) or K(G17) in the block. When the cycle is started, the control assumes that the tip of the tool is located at that distance from the surface of the workpiece. While the pocket is being milled, as soon as the material of a level is removed, the tool will be lifted to that distance so that it can be brought to the start point for milling the next level.

D = address of register containing the radius compensation of the tool

The radius-compensation number of the tool, used in the program is to be specified mandatorily at address D. Besides, the milling of a pocket has to be carried out in state G40.

E = width of cutting, in percent of milling diameter
with + sign, machining in counter-clockwise sense,
with – sign, machining in clockwise sense.

Two types of information can be specified at address E. The value of E defines the width of cutting in percent of milling diameter. Unless it is specified, the control will automatically assume +83%. The control can modify the data specified at address E, depending on the width of pocket, in order to obtain a uniform cutting in milling a particular level. Such a modification may, however, be a reduction only. The sign of address E will define the direction of milling. When E+, i.e., it is positive, the machining will be carried out in counter-clockwise sense, if it is E–, i.e., negative, the machining will be carried out in clockwise sense.

Q = depth of cut

The depth of cut can be specified at address Q in the applicable units of measure (mm or inches). Depending on the depth of pocket, the control may override the program value in order to obtain a uniform distribution of cuts. Such a modification may, however, be a reduction only.

F = feed

The feed applied in the cycle can be specified at address F. Unless F is given a value, the modal F value will be adopted. 50% of the F value will be applied

– when a level begins to be milled, and a depth of cut (Q) is drilled,

– when milling the pocket longitudinally as long as the Q is loaded on both sides.

M S T = function

A function M, S, T can be specified in the block calling the procedure of pocket milling, which will be executed by the control prior to commencement of milling.

Degenerated cases of cavity milling:

Unless the width of pocket has been specified, the radius of the pocket's corners will be taken twice for the width of pocket.

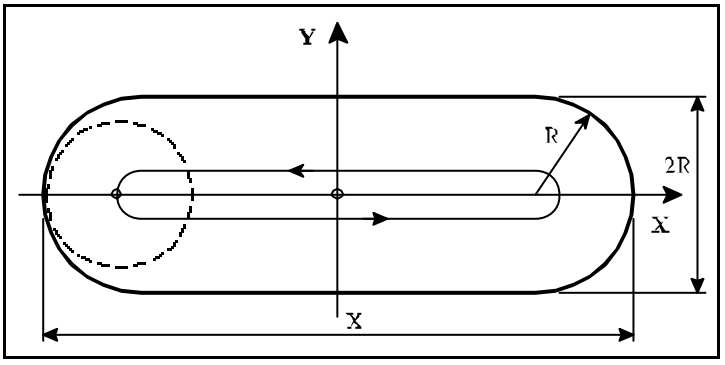


Fig. **20.18-2**

Unless the width of pocket and the rounding radii of corners have been specified, the tool diameter applied will be taken for the width of pocket (groove).

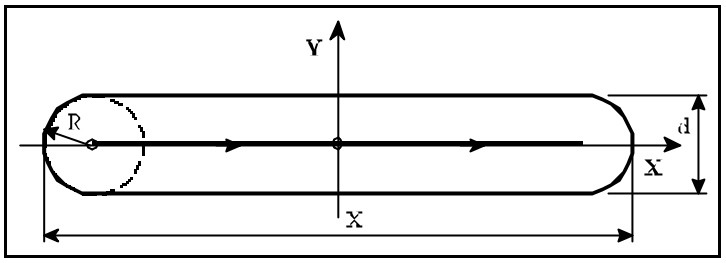


Fig. **20.18-3**

If neither the length nor the width of pocket has been specified, only address R has been programmed, a circular pocket of radius R will be milled.

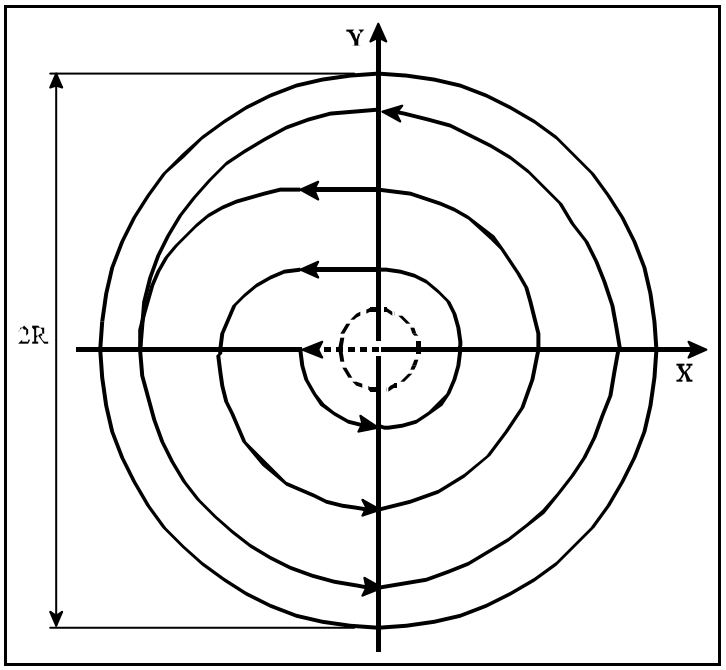


Fig. **20.18-4**

If neither length, nor width, nor radius have been specified, the cycle will "degenerate" into drilling.

Error messages in the course of pocket milling:

MACRO ERROR 1 - false block specification. Possible causes:
– Depth of pocket not specified
– Radius of tool not specified
– Depth of cut not specified.

MACRO ERROR 2 - definition error in sizes specified. Possible causes:

– The size specified for the length or width of pocket is smaller than twice of the pocket radius.
– The length or width of pocket is smaller than the diameter of tool called at address D.
– The value specified for the width of cutting is 0 or the tool radius called is 0
– The value of depth of cut is 0, i.e. 0 has been programmed at address Q.

Notes

Index in Alphabetical Order: