NUM 720 T

Programming Manual

En-938697/C

MOD. 06-92
ED. 09-88

www.num.com

CNC HighEnd Applications

Despite the care taken in the preparation of this document, NUM SA cannot guarantee the accuracy of the information it contains and cannot be held responsible for any errors therein, nor for any damage which might result from the use or application of the document.

The physical, technical and functional characteristics of the hardware and software products and the services described in this document are subject to modification and cannot under any circumstances be regarded as contractual.

The programming examples described in this manual are intended for guidance only. They must be specially adapted before they can be used in programs with an industrial application, according to the automated system used and the safety levels required.

June 1992

NUM 720 T

PROGRAMMING MANUAL

Amendment list C

Conforms to software 720 T N° 201281 index E

IMPORTANT :

Some functions described in this document may well not be offered by your system, if the particular option does not form part of the installed configuration, or if the function does not exist in that version of the product (cf version E in particular).

Modified pages :

0.1 to 0.4
2.2 to 2.6
4.2
5.11
6.14
7.4.1 – 7.4.2 – 7.6
8.3 – 8.4 – 8.16
A1.1 – A1.8 – A1.9 – A1.14 – A1.16

Additional pages :

0.5 – 0.6
6.15 – 6.16
8.3.0 – 8.3.1
8.17 to 8.20
9.15 – 9.16
11.1 to 11.4
12.1 to 12.10
A1.17 – A1.18

CONTENTS

NOTES

1 - GENERAL

1.1 - ISO CODE (NFZ 68010) - EIA CODE (RS244.B)

- Programming is of the variable, address type.

- Programming is possible by means of two codes :
  . ISO code, according to standards NFZ 68010, 68030, 68032
  . EIA code, according to standards RS244.B and 273.A.

Characters recognized

MEANING	ISO	EIA
Digits	from 0 to 9	from 0 to 9
Letters of the alphabet	A to Z	A to Z
Start of program	%	EOR
Start of comment	(	,
End of comment	)	%
Plus sign	+	+
Minus sign	-	-
Decimal point	.	.
Greater than	>	
Less than	<	
Multiplication	*	
Equals	=	
Division	/	
Commercial A	@	
End of block	LF	EOB
Block jump	/	/
End of tape	X OFF	BS

NOTE :

As comments are not provided within the framework of the EIA standards the characters "," and "%" have been retained and have the same effect on the system as the parentheses in ISO.

IMPORTANT :

As the equivalent of the characters >, <, *, = and @ are not defined in the EIA code, parameterized programming is forbidden in this code.

Characters which are recognized by the system but have no effect on its operation

MEANING	ISO	EIA
Tabulation Carriage return Space Error	HT CR SP DEL RUB OUT	TAB SP DEL RUB OUT

NOTE :

In EIA code, a missing character is indicated by a parity error.

Punching combinations for special characters in ISO code

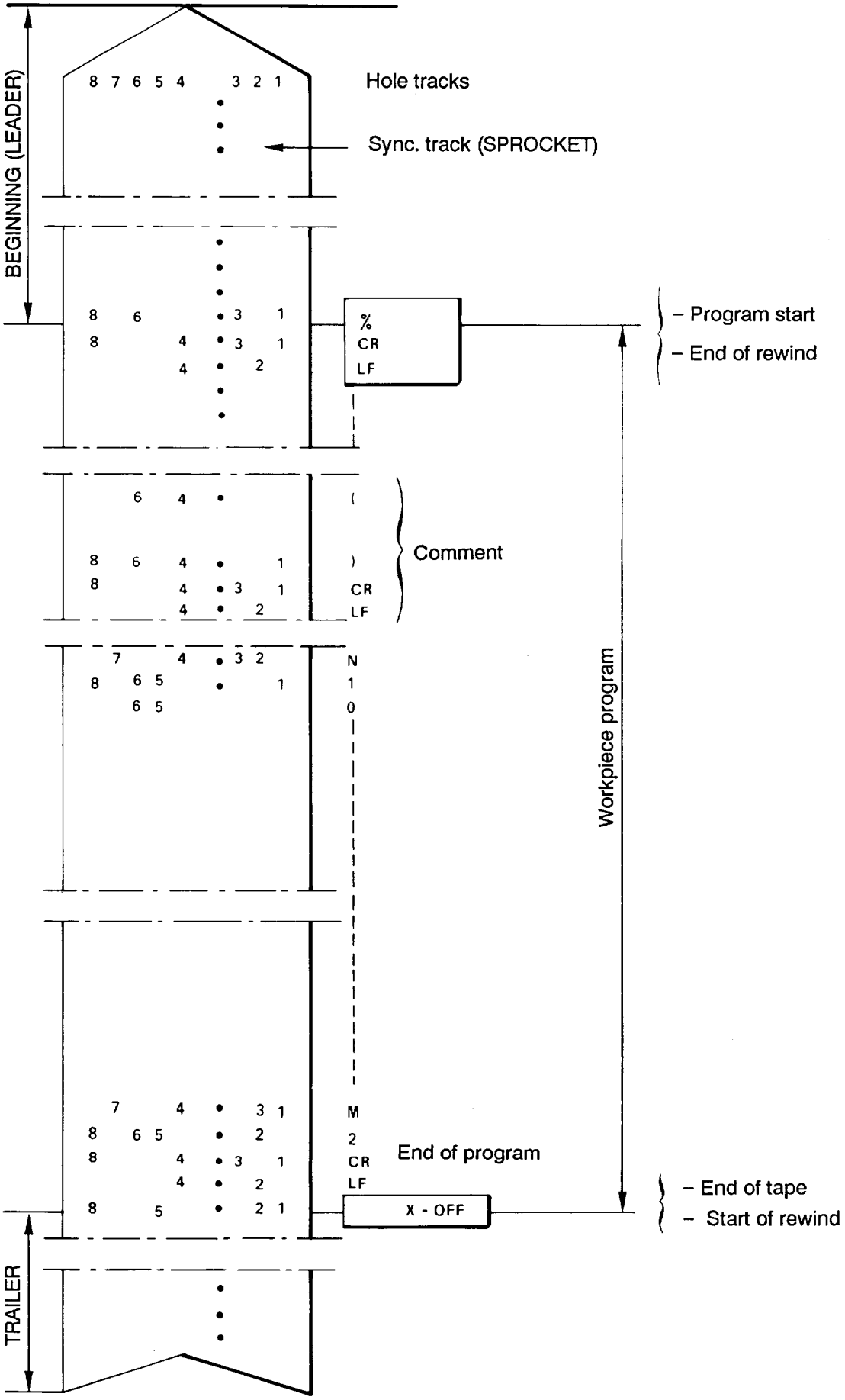
No. of data tracks		8	7	6	5	4		3	2	1
Meaning	Character	Punching combinations								
Less than	(			●	●	●	·	●		
Greater than	)	●		●	●	●	·	●	●	
Multiplied	*	●		●		●			●	
Equal	=	●		●	●	●	·	●		●
Division or block jump	/	●		●		●	·	●	●	●
Commercial A	@	●	●				·			
AND	&	●		●			·	●	●	
OR	!			●			·			●
Dollar	$			●			·	●		
Comma	,	●		●		●	·	●		
Decimal point	.			●		●	·	●	●	
Apostrophe	'			●			·	●	●	●
Semi-colon	:	●		●	●	●	·		●	●
Hash	#	●		●			·		●	●
Question mark	?			●	●	●	·	●	●	●
Inverted commas	"			●			·		●	

ISO CODE : NFZ 68010 – EIA CODE : RS244.B

		ISO								
Hole tracks		8	7	6	5	4		3	2	1
Meaning	Character	Hole								
Prog. start, rewind stop	%	●		●			●	●		●
+ sign	+			●		●	●		●	●
– stop	–			●		●	●	●		●
Digits	0			●	●		●			
	1	●		●	●		●			●
	2	●		●	●		●		●	
	3			●	●		●		●	●
	4	●		●	●		●	●		
	5			●	●		●	●		●
	6			●	●		●	●	●	
	7	●		●	●		●	●	●	●
	8	●		●	●	●	●			
	9			●	●	●	●			●
Angular coordinates (assoc. with E)	A		●				●			●
Angular coordinates (assoc. with E)	B		●				●		●	
Angular coordinates about Z axis	C	●	●				●		●	●
Tool compensator	D		●				●	●		
External parameter	E	●	●				●	●		●
Feedrate or Dwell	F	●	●				●	●	●	
Preparatory function	G		●				●	●	●	●
Sub-program no.	H		●			●	●			
Interpolation address	I	●	●			●	●			●
Interpolation address	J	●	●			●	●		●	
Interpolation address	K		●			●	●		●	●
Program variable	L	●	●			●	●	●		
Auxiliary function	M		●			●	●	●		●
Block number	N		●			●	●	●	●	
	O	●	●			●	●	●	●	●
Various parameters	P		●		●		●			
	Q	●	●		●		●			●
	R	●	●		●		●		●	
Spindle speed	S		●		●		●		●	●
Tool number	T	●	●		●		●	●		
Secondary axis // to X	U		●		●		●	●		●
Secondary axis // to Y	V		●		●		●	●	●	
Secondary axis // to Z	W	●	●		●		●	●	●	●
Primary axis. X	X	●	●		●	●	●			
Primary axis. Y	Y		●		●	●	●			●
Primary axis. Z	Z		●		●	●	●		●	
Program subdivision	:			●	●	●	●		●	
Optional block	/	●		●		●	●	●	●	
Carriage return	CR	●				●	●	●		●
Line feed. (End of block)	LF					●	●		●	
Start of comment	(			●		●	●			
End of comment	)	●		●		●	●			●
Space	SP	●		●			●			
End of tape character	XOFF	●			●		●		●	●
Horizontal tab.	HT					●	●			●
Delete	DEL	●	●	●	●	●	●	●	●	●
Null	NUL						●			

	EIA								
	8	7	6	5	4		3	2	1
Character	Hole								
EOR					●	●		●	●
+		●	●	●		●			
–		●				●			
0			●			●			
1						●			●
2						●		●	
3				●		●		●	●
4						●	●		
5				●		●	●		●
6				●		●	●	●	
7						●	●	●	●
8					●	●			
9				●	●	●			●
a		●	●			●			●
b		●	●			●		●	
c		●	●	●		●		●	●
d		●	●			●	●		
e		●	●	●		●	●		●
f		●	●	●		●	●	●	
g		●	●			●	●	●	●
h		●	●		●	●			
i		●	●	●	●	●			●
j		●		●		●			●
k		●		●		●		●	
l		●				●		●	●
m		●		●		●	●		
n		●				●	●		●
o		●				●	●	●	
p		●		●		●	●	●	●
q		●		●	●	●			
r		●			●	●			●
s			●	●		●		●	
t			●			●		●	●
u			●	●		●	●		
v			●			●	●		●
w			●			●	●	●	
x			●	●		●	●	●	●
y			●	●	●	●			
z			●		●	●			●
0		●				●	●	●	
/			●	●		●			●
						●			
EOB	●					●			
?			●	●	●	●		●	●
%		●		●	●	●		●	●
SP				●		●			
BS			●		●	●		●	
TAB			●	●	●	●	●	●	
DEL		●	●	●	●	●	●	●	●
NUL						●			

STRUCTURE OF THE PROGRAM TAPE (Example in ISO)



1.2 – PROGRAM STRUCTURE

1.2.1 – Workpiece program

- The tape must begin with the character :
  . % in ISO
  . EOR in EIA (end of record).

The ISO – EIA code is recognized by the system on reading this character.

- The tape must end with the character :
  . X OFF in ISO
  . BS in EIA (Backspace).

FORMAT OF THE CONTROL BLOCKS

- Blocks with variable, address format.

- Decimal point : if the unit used is the millimeter, .01 represents 1/100 mm

- It is possible to omit:
  . the "+" sign, taken by default for dimensions,
  . leading and trailing zeros, (eg 0.010 or .01),
  . spaces, tabulations and RUB OUT.

The manufacturer may write sub-programs in the EEPROM memory (see manufacturer's handbook). The available memory capacity is 10 Kbytes.

The maximum memory capacity for a single workpiece program is 64 Kbytes.

1.2.2 – Tool compensations on a separate tape

ISO Format

% (comment of upto 40 characters)				CR	LF		
D1	X ± 4.3	Z ± 4.3	R ± 3.3	C1	CR	LF	(C1 = tip orientation)
D2	X ± 4.3	Z ± 4.3	R ± 3.3	C1	CR	LF	
.							
.							
.							
D32	X ± 4.3	Z ± 4.3	R ± 3.3	C1	CR	LF	
XOFF	LF						

NOTE :

The tool compensation tape can be in ISO or EIA code.

NOTES

2 - PROGRAMMING

2.1 - GENERAL DATA FORMATS AND ADDRESS DEFINITIONS (ISO/DIS 6983/1 APPENDIX C)

Program dimensions which may contain a decimal part are expressed thus:

X + 36 is equivalent to X = 36 mm
X - .3 is equivalent to X = -0.3 mm

% 04	Program number
N05	Sequence number (0 to 32767)
G02	Preparatory functions. (See list)
H04	Sub-program number, used with G77
X+053	X axis demands, programmed on diameter or radius
Z+053	Z axis demands
I+053	In G2 or G3, absolute or incremental coordinates of the centre of the circle In G64 or G65, the stock allowance in X
K+053	In G33 or G38, the thread pitch in X (or Z if m/c'g a scroll). In G64 or G65, the stock allowance in Z
EA+033	In G1, the angle, in degrees, of a straight line with respect to the Z axis In G33, the angle of the taper.thread In G65, the angle of penetration, used to rough out a re-entrant groove In G66, the angle of the bottom of the groove
EB+053	EB+ in G1, G2 or G3, defines a blend radius, between any two elements EB- in G1, defines a chamfer between two straight lines EB33 in G33, defines the flank penetration angle
C033	With M19, gives 360° of spindle orientation
P053	In G33, the total thread depth In G64 or G65, the penetration of each cut in the X direction In G83 or G87, the value of the first penetration
Q053	In G33, the depth of the last cut In G65, the next cut, rapid approach value In G83 or G87, the value of the last penetration
R053	In G2 or G3, the radius of a circle In G33, the tapered pullout distance, along X or Z In G64 or G65, the penetration of each cut, in the Z direction In G66, the pitch, in Z In G92, radius of the reference circle for application of tangential feed
R±	In G0 or G1, positioning at a programmed distance from a point

F052	In G94, the feed rate expressed in mm/min ; minimum 0.01 mm/min In G95, F023 : the feed rate in mm/rev. Maximum 16 mm/rev. In G33, F01 : the number of thread starts In G04, F022 : the length of dwell in seconds, max 99.99 secs
EF022	In G65, the penetration feedrate In G66, the dwell at the bottom of the groove In G83 or G87, the dwell at the end of each penetration
EF05	Special feed rate for a fillet or chamfer
EG03	Programmed acceleration override (1–100%)
EX053 EZ053	In G23, intermediate point on a circle In G23, intermediate point on a circle
EK053	Spindle in/out speed ratio for rigid tapping
M03	Auxiliary functions : 22 decoded, 242 coded. (See list)
S05	In G97, the spindle speed in rpm In G96, the cutting speed in m/min In G92, the maximum spindle speed in rpm In G33, the number of cuts In G77, the number of repetitions of a sub–program, S04
T03	Tool number from 0 to 255
D02	Compensation number from 0 to 32
L03	Program variables from 0 to 19, 100 to 199 and from 900 to 939
E113	External parameters

2.2 – G FUNCTIONS

CODE	CANCELLATION	DESCRIPTION	SEE §
G00	G01–02–03–23–33..	Linear interpolation at rapid	4.1.1
G01*	G00–02–03–23–33..	Linear interpolation at program feed	4.1.1
G02	G00–01–03–23–33..	Clockwise circular interpolation at tangential feedrate	4.1.2
G03	G00–01–02–33..	Identical to G02, but in anti–clockwise direction	4.1.2
G04	End of block	Dwell, programmable with the F address	7.2
G09	End of block	Precise stop at end of block	7.1.3
G10	End of block	Premature block termination, with optional block jump	9.2
G12	End of block	Handwheel feedrate enhancement	7.1.5
G16*	End of block	Direction of tool axis, using P,R addresses	6.5
G23	G00–01–02–03–33	Programming of a circle by three points	4.1.2
G33	G00–01–02–03	Constant pitch threading	6.6
G38	G00–01–02–03	Sequence of threading blocks	6.7
G40*	G41 – G42	Tool radius compensation cancellation	5.7
G41	G40 – G42	Tool radius compensation to the left of the profile	5.7
G42	G40 – G41	Tool radius compensation to the right of the profile	5.7
G52	End of block	Absolute programming with respect to the machine origin	3.2
G53	G54	Suspension of DAT1 and DAT2 datum shifts	
G54*	G53	Enabling of DAT1 and DAT2 datum shifts	
G59	End of block	Programmed datum shift, added to DAT1 and DAT2 in G54	3.1.3
G64	G80	Turning/facing area clearance cycle	6.1
G65	End of block	Recessing area clearance cycle	6.2
G66	End of block	Grooving area clearance cycle	6.3
G70	G71	Data input in inches	9.8
G71*	G70	Data input in metric	
G75	End of block	Emergency withdrawal, sub–program definition	9.4

CODE	CANCELLATION	DESCRIPTION	SEE §
G77	End of block	Sub-program call, with a return	9.3.1
G79	End of block	Conditional or unconditional jump, without a return	9.3.2
G80*	G64-83-87	Machining cycle, cancel code	
G83	G80-64-65 G66-87	Woodpeck drilling cycle	6.4
G84	G80-64-65 G86-83-87	Rigid tapping	6.8
G87	G80-64-65 G66-83	Chip-break drilling cycle	6.5
G90*	G91	Absolute programming	3.2
G91	G90	Incremental programming	3.2
G92 R	G92 R0 - M02	Programming of tangential feed	7.1.6
G92 Sxx	M2	Spindle speed safety limit	7.3.3
G92 XorZ	End of block	Program origin pre-selection	3.2.4
G94*	G93-95	Feed rate expressed in mm/min	7.1.1
G95	G93-94	Feed rate expressed in mm/rev	7.1.2
G96	G97	Constant cutting speed	7.3.2
G97*	G96	Spindle speed in rpm (automatic range selection)	7.3.1

* Indicates the default condition. ie set when the system is switched on, or after a reset.

2.3 – DECODED M FUNCTIONS

CODE	FUNCTION		CANCELLATION	DESCRIPTION	SEE §
	BEFORE	AFTER			
M00		X	Cycle start	Programmed stop	
M01		X	Cycle start	Optional stop	
M02		X	% or EOR	End of workpiece program	
M03	X		M4–M5–M0–M19	Clockwise spindle rotation	
M04	X		M3–M5–M0–M19	Anti–clockwise spindle rotation	
M05*		X	M3–M4	Spindle stop	
M06		X	Report	Tool change	
M07	X		M9–M2	Coolant No.2, on	
M08	X		M9–M2	Coolant No.1, on	
M09*		X	M7–M8	Coolant off	
M10		X	M11	Axis clamp	
M11	X		M10	Axis unclamp	
M12		X	Cancellation of FEED–STOP	Setting–up of "RECALL" mode	9.14
M19		X	M3–M4–M5	Spindle orientation	4.1.3 7.4
M40 to M42	X			3 spindle ranges	
M48*		X	M49	Enables the speed and feed potentiometers	
M49	X		M48	Disables the speed and feed potentiometers	
M997	X		M998–M999–M2	Forces block continuation	9.10
M998*	X		M999–M997	Allows EDIT and MDI modes and sub–program calls by the PLC	8.1.2
M999	X		M997–M998 M2	Disallows EDIT and MDI modes and PLC sub–program calls	8.1.2

* Indicates the default conditions, ie when the system is switched on, or after a reset.

NOTE :

- *Only M6 is non modal. It is reset to zero as soon as the M acknowledgement is detected by the CNC. (CRM)*
- *Several decoded M functions (eg M3, M8, M41) can be programmed in one block.*
- *Other "coded" M functions, as defined by the manufacturer, are "after" functions. Only one of these functions is allowed in a block.*

2.4 – THE ADDRESS EQUIVALENCE TABLE

This table is used to obtain an equivalence between one address and another. It allows a program to be written for one axis, eg X, and makes it possible to use the same program for another axis, eg Z.

The @ symbol followed by an address (A to Z) designates an equivalent address.

An equivalent address is declared by programming :

@ { A ' ' Z } = { A ' ' Z }

A value is assigned to an equivalent function by programming

@ { A ' ' Z } Followed by the value

EXAMPLE : @ X = Z

@ X 300 is then equivalent to Z 300

The table of equivalence of addresses is initialized when the system is switched on, or by reset or M02.

(@ A = A @ B = B ... @ Z = Z).

Declaration of a new equivalent address delays preparation of the block in which it is programmed until the preceding block has been executed.

The table of address equivalence is displayed by pressing the "Program variables" key, which operates as a bistable. ie. Press once to display the program variables, press again to display the table of address equivalence.

See example in paragraph 8.5.

3 – DIMENSION PROGRAMMING SYSTEMS

3.1 – DEFINITIONS

The controller always processes dimensions with reference to a DATUM or measurement origin, in accordance with the programming mode chosen (G90, G91, or G52).

3.1.1 – Origins

The origins are as follows :

- M/C measurement origin : Om

  This is a reference point defined on each axis by the machine manufacturer and is usually established by contacting a switch.
  It is used to establish the absolute measurement origin.

- Workpiece origin : Ow

  This origin is independent of the measurement system and is set at a known point, perhaps on a location fixture. From this known point, the workpiece may be positioned directly or may be offset, by using shims or a comparator.

- Program origin : Op

  The program origin is independent of the measurement system. It represents the origin of the reference coordinate system from which the programmer has chosen to write his part program.

3.1.2 – DAT 1

After referencing the m/c, (to establish Om), the initial display shows the X and Z dimensions pre–assigned to the axes by the m/c builder. The operator then establishes the workpiece origin as follows:–

- After physically jogging the slides to the required positions, the "current position" values can be entered semi–automatically.
  * X Z LF is input in DAT 1 mode. The displayed position will then zero, and will correspond to the absolute value of the current position with respect to the machine origin (MOS).

- If these values are already known, then they can be entered directly through the keyboard, in DAT 1 mode.

3.1.3 – Other offsets

- DAT 2

  This offset is always added to DAT 1. It is available simply for the sake of convenience, to hold a shim thickness for example, in order to more easily set up the workpiece origin.

  DAT 2 is often known by the programmer, who can then instruct the operator, (via a set up sheet,) into what value to enter through the keyboard. It is taken into account by the system in all cases except when in G52 or G53.

- A PROGRAMMED OFFSET, eg. G59 X.. Z..

  From the workpiece origin, a programmed offset may be used to re–position the part program origin, (Op), to any point. This is to allow for easy interpretation of the part drawing dimensions, and to simplify the programming of repetitive elements.

  . In absolute programming (G90)

    G59 Xn defines the value of the offset to be applied to all X dimensions that follow. A new G59 Xn cancels and replaces the previous one, and all subsequent X axis demands take the value of the last G59 programmed into account.

  . In incremental programming (G91)

    G59 Xn defines the value of the offset to be applied to the first X dimension after the G59. A new G59 will, in the same way, modify the first dimension which follows it. Thus, the absolute value of the positions will be modified by the sum of all the G59s previously programmed on that axis.

NOTE :

To cancel the offset or offsets applied in G59, it is recommended :

- *in absolute programming (G90) :*
  to program G59 X0 Z0.

- *in incremental programming (G91) :*
  to revert to absolute programming (G90) and program G59 X0 Z0.

3.2 – RULES FOR THE USE OF OFFSETS

Programmed axis demands are always expressed from an origin.

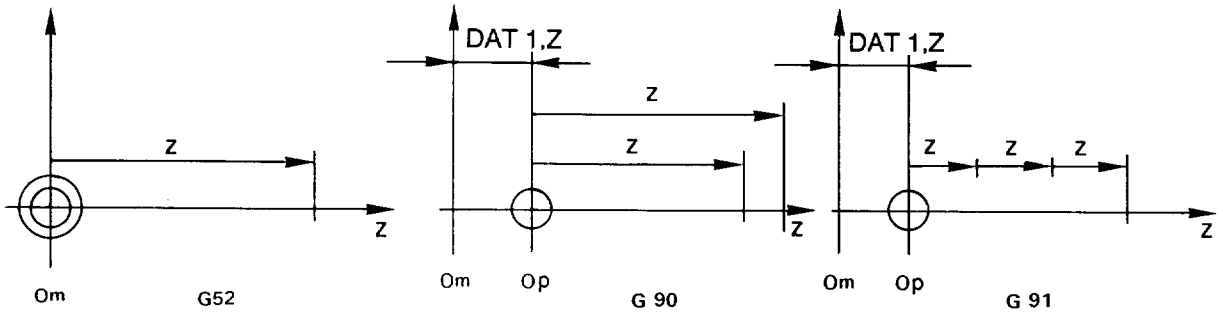
The dimensions programmed may be given as follows :

. Absolute programming (G90)
  The dimensions are taken with respect to the program origin, Op.

. Incremental programming (G91)
  The dimensions are taken with respect to the previous position.

. Absolute programming (G52)
  The dimensions are taken with respect to the measurement origin, Om.

With G52, no offsets or tool compensations are taken into account. This function is cancelled at the end of the block. It must be programmed before the axes addresses and without radius compensation (G41 – G42). This programming mode is used to position an axis to a fixed point with respect to the machine structure. (eg for automatic tool change sequences).



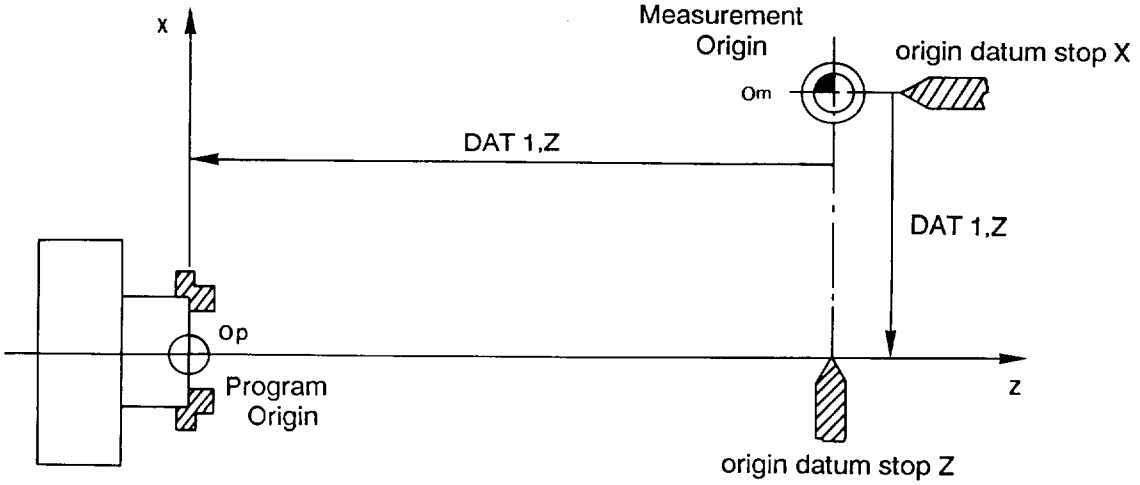
NOTE :

- *The first program movement along each axis MUST be programmed in absolute mode (G90) Otherwise, G91 will be taken as a G90 for this movement. No error will be reported!*
- *All programmed dimensions, modified by the various offsets and tool dimensions, are compared with the axes limits as defined by the builder. If these dimensions are exceeded, an error will be reported.*

3.2.1 – DAT 1 and DAT 2

As previously described, DAT 1 is set with respect to the m/c measurement origin by positioning the tool to a datum point on the workpiece (Ow), and entering that position in DAT 1 mode.

DAT 2 represents any distance between the point at which DAT 1 was set by the operator, and the program origin (Op) as required by the programmer. It may not be necessary to enter any value in DAT 2, in which case Ow and Op are coincident



Once DAT 1 (and DAT 2) have been entered, the program origin (Op) will be in accordance with the workpiece program tape.

NOTE :

– The DAT 1 and DAT 2 values are enabled by G54. They are disabled by G53.

3.2.2 – Offsets programmed by G59

Program origin offsets must not exceed the system measurement capacity.

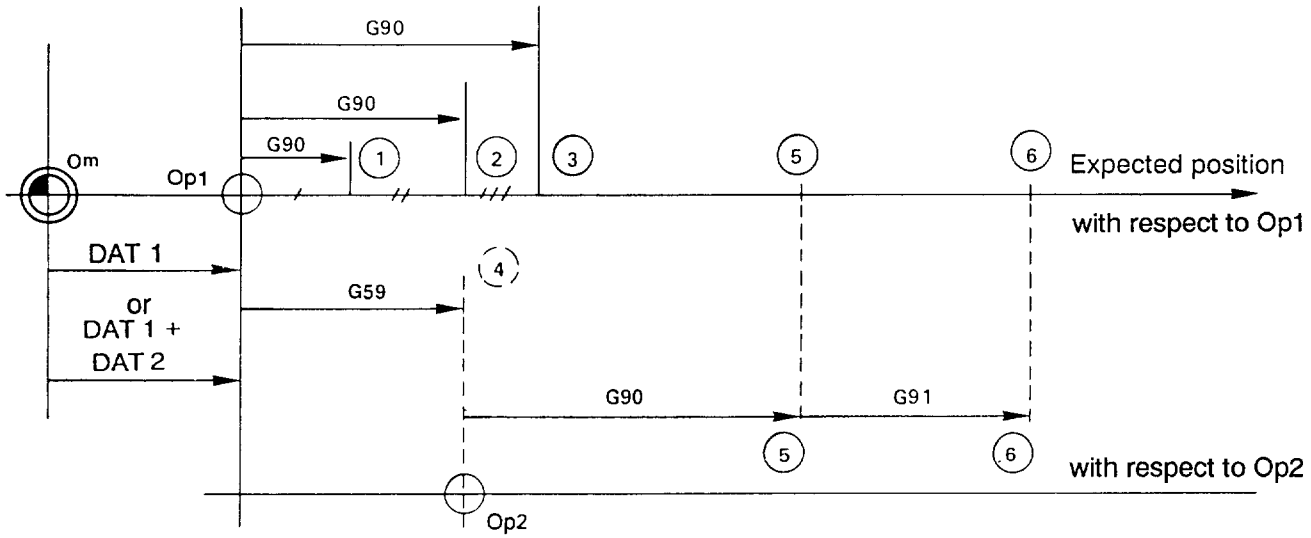
An offset programmed with G59 is modal.

eg. G59 X ± 5.3 Z ± 5.3

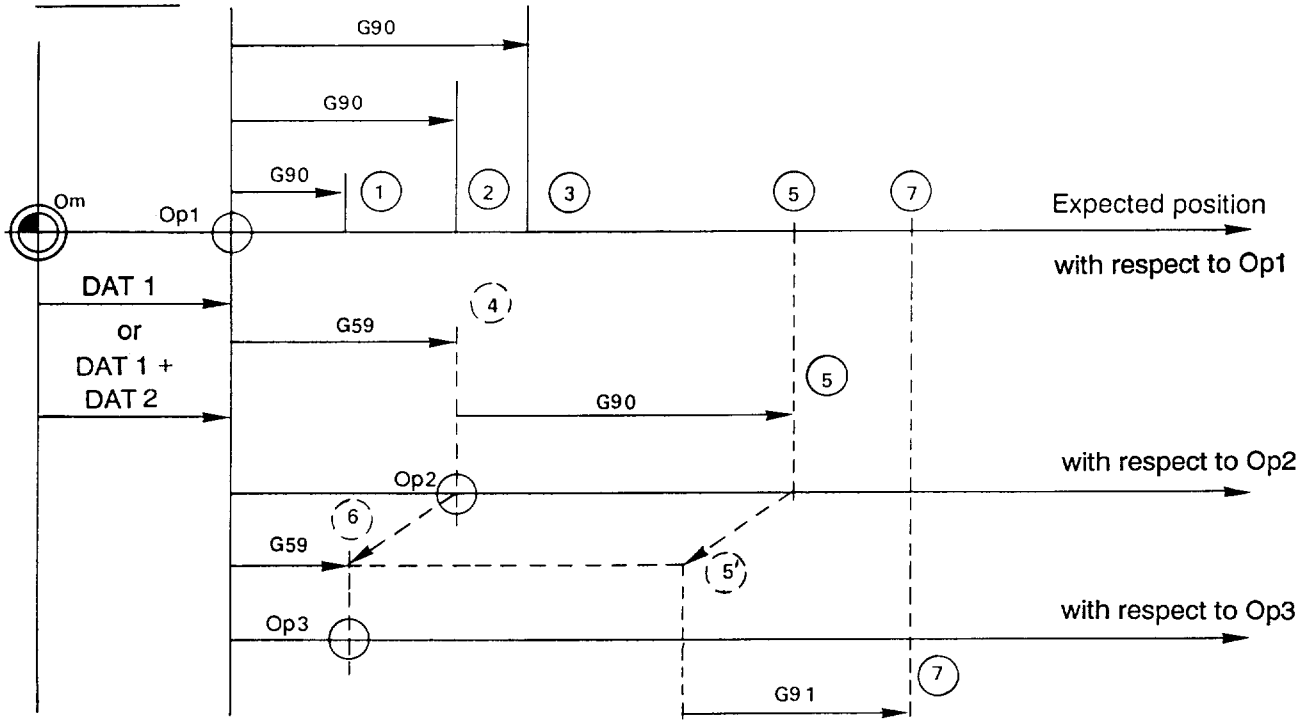
There are no axis movements with G59, simply a relocation of the origin.

Absolute programming (G90 – G59)

EXAMPLE 1



EXAMPLE 2



When the system is programmed in absolute mode, (G90), a G59 offset is added to the program origin as defined by DAT 1 + DAT 2.

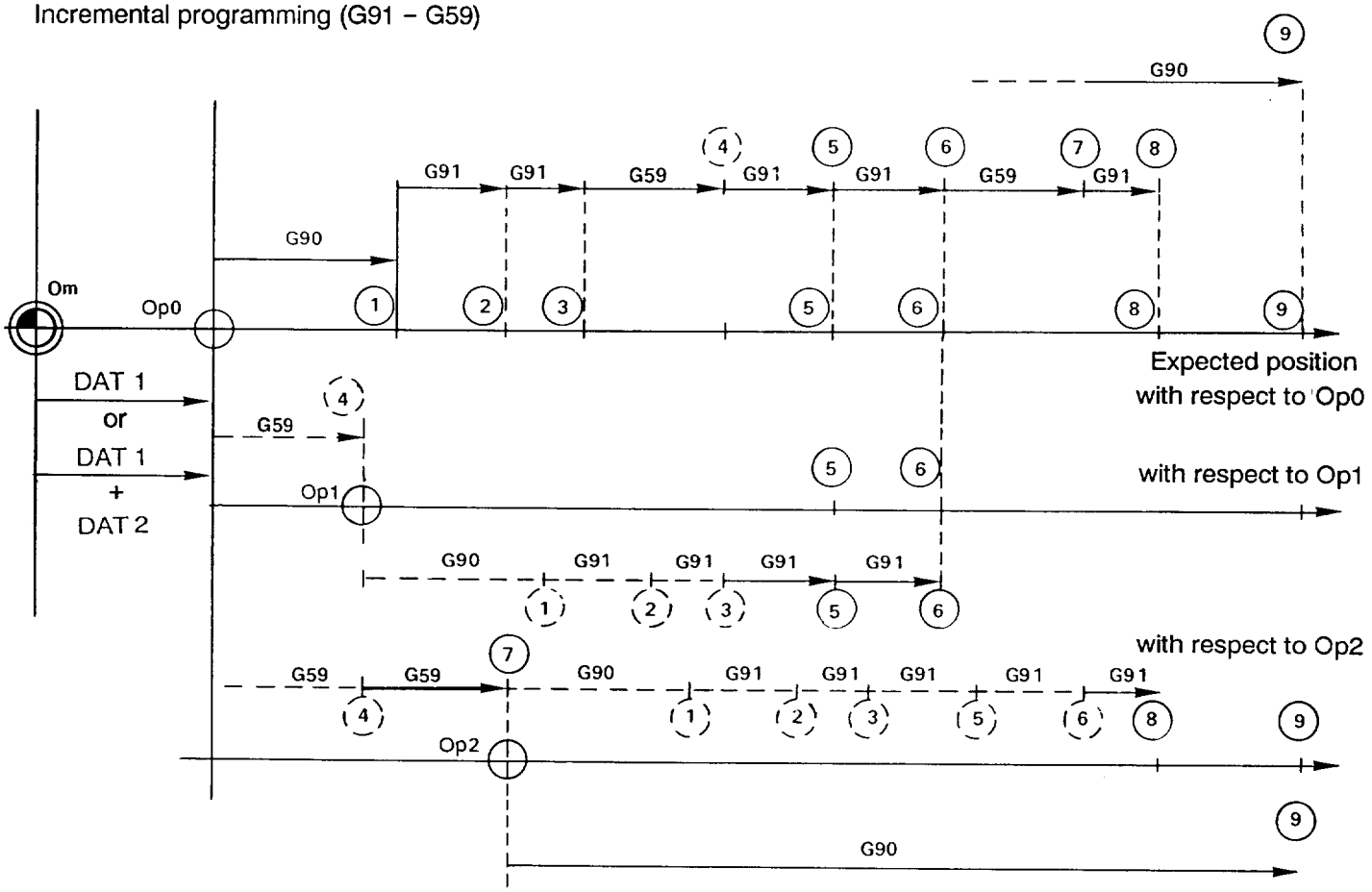
EXAMPLE 1 :

The absolute dimensions programmed after G59 are translated by the value programmed with G59 (point 5). Any incremental values programmed after a G90, (point 6), are with respect to the previous point.

EXAMPLE 2 :

The reasoning used in the first example applies up to point 5. A second G59 then "translates" the coordinate system without axis motions.
Point 7 will therefore be defined with respect to the new point 5' and not the old point 5.
Programmed origin offsets can be removed by programming a zero offset on the translated axes, eg : G90 G59 X0 Z0.

Incremental programming (G91 - G59)



When the system is programmed in incremental mode (G91), the first dimension programmed after G59, (point 5), is translated by the value of the offset. The absolute origin Op1 is determined from Op0 using the same value.

If a second G59 origin offset is programmed in the same way, the new absolute origin Op2 is determined from Op0 by a translation equal in value to the sum of the two offsets.

On returning to absolute (G90) programming, (point 9), the absolute dimension will now be defined with respect to Op2.

Incrementally programmed origin offsets may be removed by programming :

- either the offset corresponding to the sum of all the offsets carried out but with opposite sign, if still in incremental mode,

- or a zero origin offset, after switching to absolute programming : G90 G59 X0 Y0 Z0.

NOTE :

- *A manual reset or (MO2 reset) returns the system to its initial status (Op0).*

- *To simplify and understand a part program, it is strongly recommended to be in, G90 status before programming an origin offset (G59). Simply programming G59 X0 Z0 will then return to the initial reference system.*

- *Programming in m/c measurement origin mode (G52) is allowed only in G40 mode. Otherwise, the system will report error 27.*

3.2.3 – Preselection of the program origin : G92 X or Z

The dimensions programmed after the non–modal function G92, define the current position of the slide with respect to the program origin. DAT 1 is automatically updated as follows:

DAT 1 = machine dimension – programmed dimension – tool length compensation.

Note that DAT 2 is not affected by the above equation.

G92 has the effect of closing down the program "look ahead" decode, so preventing its use in radius compensation mode (G41 or G42), or any other mode which requires "look ahead" to be active.

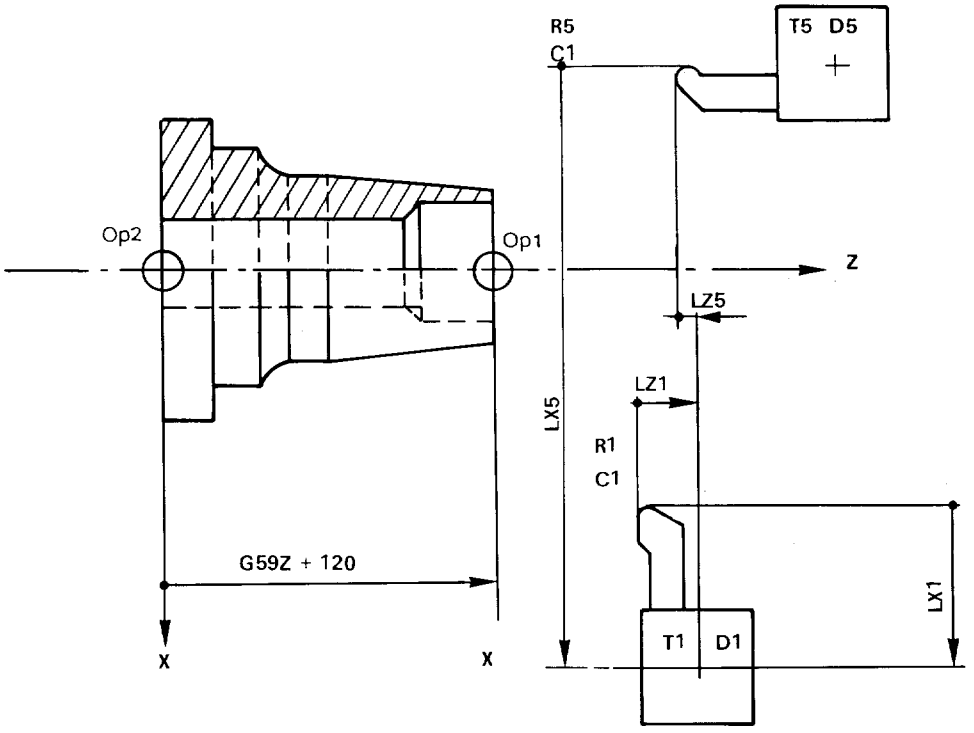
NOTE :

- *G92 is not executed in TEST or SEARCH modes, so do NOT search to a block following a G92.*

- *The new DAT 1 is retained at the end of the program and so can affect the next part program, or even the following part, using the same program. (The main use of G92 is to establish a datum after probing some feature of the job.)*

- Examples of the use of an origin offset

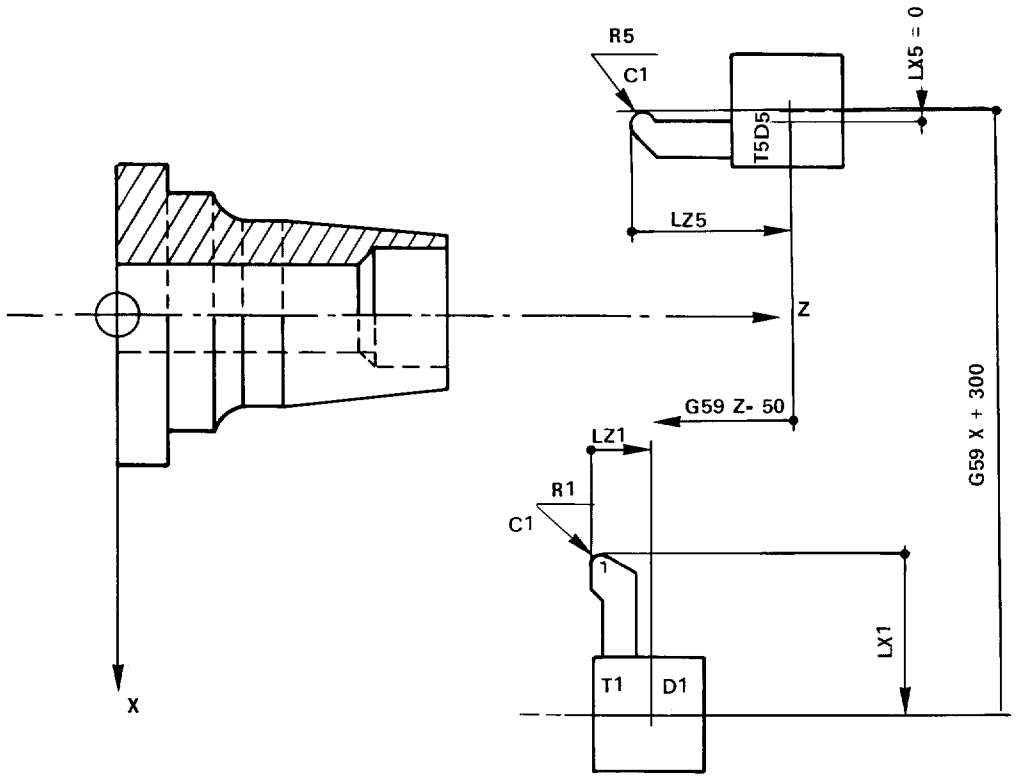
EXAMPLE 1 :



External machining from the rear turret with (Op0) as the program origin followed by internal machining from the front turret, with (Op1) as the origin.

N20		D_1	M6	External machining origin Op0
N150		D_5	M6	
N160	G59	Z120		Internal machining origin Op1
N280			M2	

EXAMPLE 2 :



On a twin turret lathe (see 5.2.3.), all tool dimensions are defined in relation to the main turret.

However, the tool dimensions of the secondary turret can be defined in relation to the reference point of the secondary turret by programming an offset (G59) when that turret is to be used.

N20			T1D1	M6	Machining on the front turret
N150			T5D5	M6	
N160	G59	X300	Z–50		Machining on the rear turret
N280				M2	

3.3 – RADIUS OR DIAMETER PROGRAMMING

The concept of radius or diameter programming on a lathe, affects not only the programming of the workpiece but the definition of tool dimensions and tool wear compensations, origin offsets, manual controls and the display.

Radius or diameter programming mode is fixed by machine parameter.

3.3.1 – Radius programming

In radius programming mode, everything relating to movement along the X axis is expressed as a function of the radius. Only the dynamic tool compensations are input as diameter values, for ease of entry of these measured values.

3.3.2 – Diameter programming

In diameter programming mode, not all values are expressed as a function of diameter, others must still be expressed as a function of radius :

a) Programming

- Diameter :
  . Attachment dimension, for constant cutting speed, (X)
  . Absolute dimensions (G90) : movement demands (X) and circle centres (I)

- Radius :
  . Incremental dimensions (G91) : movement demands (X) and circle centres (I)
  . Circle radii
  . Chamfer and fillet radii
  . Penetration depths for area clearance cycles
  . Stock allowance for area clearance cycles
  . Thread depths
  . Cross drilling cycle movements
  . G59 and G52 values

- Cone angle :
  . In all cases it is the half–angle at the vertex which is programmed.

b) Tool dimensions
Radius input.

c) Tool wear compensation
Diameter input, but radius display, on the tool page.

d) Origin offset
Radius input.

e) Manual jog controls
Radial movement, diameter display on the current machine position page.

4 – AXIS MOVEMENTS

The system can control two servo–controlled axes.

These axes can move in linear or circular interpolation.

Generally, the velocity of the axes is related to the spindle speed. The feed rate is therefore usually expressed in mm/rev. (G95).

Axis movement may be independent of the spindle. The feed rate is then expressed in mm/min (G94).

In threading, axis movement is very closely related to the position of the spindle. The feed rate is then defined as the thread pitch, which in turn is linked to the spindle speed.

4.1 – PROGRAMMING DIMENSIONS

4.1.1 – Linear Interpolation

G1 – linear interpolation at program feedrate. (A modal function).

This is the default condition which is automatically set when the system is switched on, or after a reset. It is mutually exclusive with the G0, G2 and G3 functions.

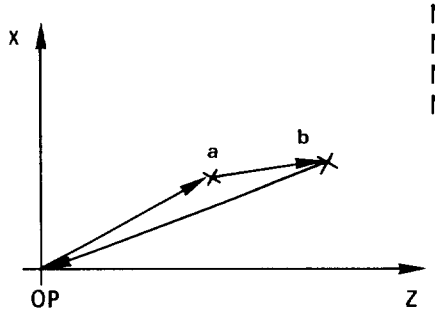
X ± 5.3 Z ± 5.3 – The maximum dimension for linear interpolation is ± 99999.999 mm.

The dimensions are expressed either in absolute programming mode, (G90), with respect to the program origin Op, (see 3.2), or in incremental programming mode, (G91), with respect to the previous end point, or in absolute programming mode with respect to the measurement origin, (G52).

NOTE :

The G0 function generates a linear movement at rapid traverse rate. This is a modal function.

EXAMPLE :



N10 G0	XOp ZOp		(rapid to Op)
N20 G1	Xa Za	F	(feed to a)
N30 G91	Xab Zab		(feed to b)
N40 G0 G90	XOp ZOp		(rapid to Op)

4.1.2 – Circular interpolation

The interpolation of a complete circle can be programmed in a single block, with I and K.

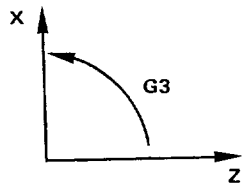
Maximum radius R 99999.999 mm.

The direction of rotation is programmed by G02 (clockwise) or G03 (anti–clockwise).

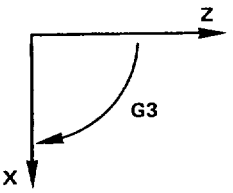
Preparatory function :

G2	G3	G23
Circular interpolation clockwise	Circular interpolation anti–clockwise	Circular interpolation direction depending on the intermediate point

They are cleared by any of the cancelling functions :
G0, G1, G3, G23, G33, G38



Rear turret



Front turret

Addresses with function G2 or G3

X ± 5.3 Z ± 5.3 { R 5.3 / I ± 5.3 K ± 5.3 }

Finishing point of the circle

Coordinates of the centre of the circle or the circle radius

The end point dimensions are also expressed either in absolute mode with respect to the program origin (Op), or in incremental mode with respect to the circle start point.

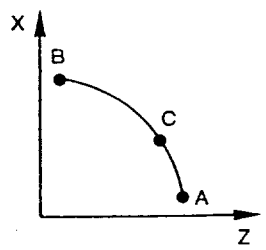
The addresses X, Z, I, K and R, MUST be programmed in the same block, even if I and K are zero or,in the case of (X or Z or R), are unchanged from previously entered values.

Addresses with function G23

EX ± 5.3 EZ ± 5.3 — Intermediate point

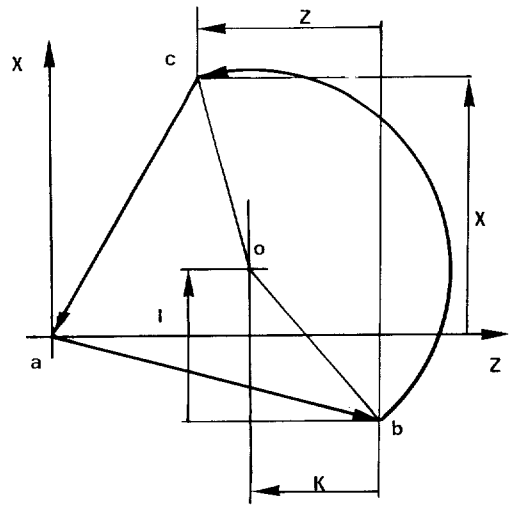
X 5.3 Z 5.3 — End point

The direction of interpolation depends on the position of the intermediate point with respect to the start point: clockwise interpolation to the left of the line pointing from start point to end point and anticlockwise interpolation to the right.



IMPORTANT :
Programming must always be in absolute dimensions.
N10 Z_A X_A Circle start point
N20 G23 EZ_C EX_C Z_B X_B
(anticlockwise circular interpolation)

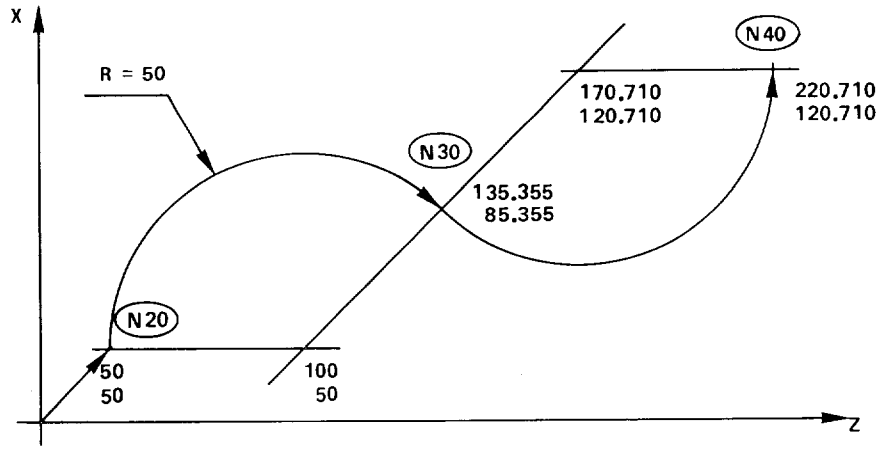
EXAMPLE : Programming a circle in incremental mode



N10 G90 G0 Xa Za
N20 G1 Xb Zb F
N30 G91 G3 X Z Ibo Kbo
N40 G90 G1 Xa Za

N10 : position to point 'a'
N20 : feed to the start point of the circle
N30 : make the circular move
N40 : cancel the G3 function

EXAMPLE : Absolute programming

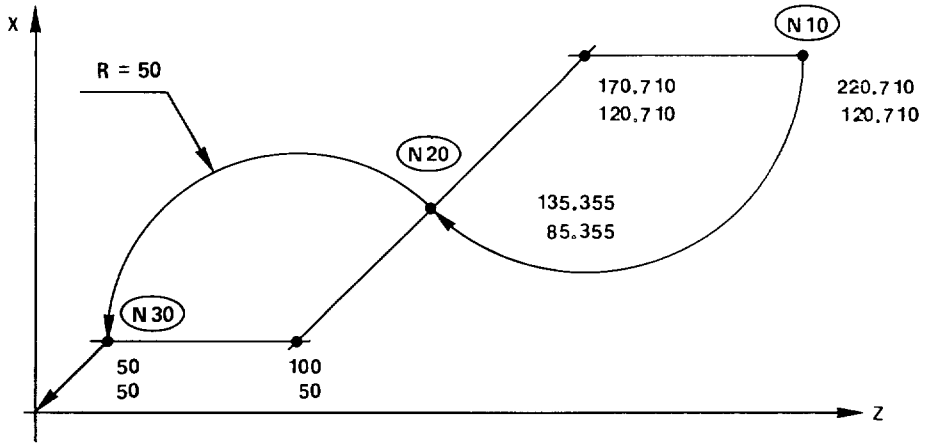


NOTE :

Both I and X are programmed on diameter in this example

N10	G0	X	Z			
N20	G1	X100	Z50			F500
N30	G2	X170.71	Z135.355	I100	K100	
N40	G3	X241.42	Z220.710	I241.42	K170.71	

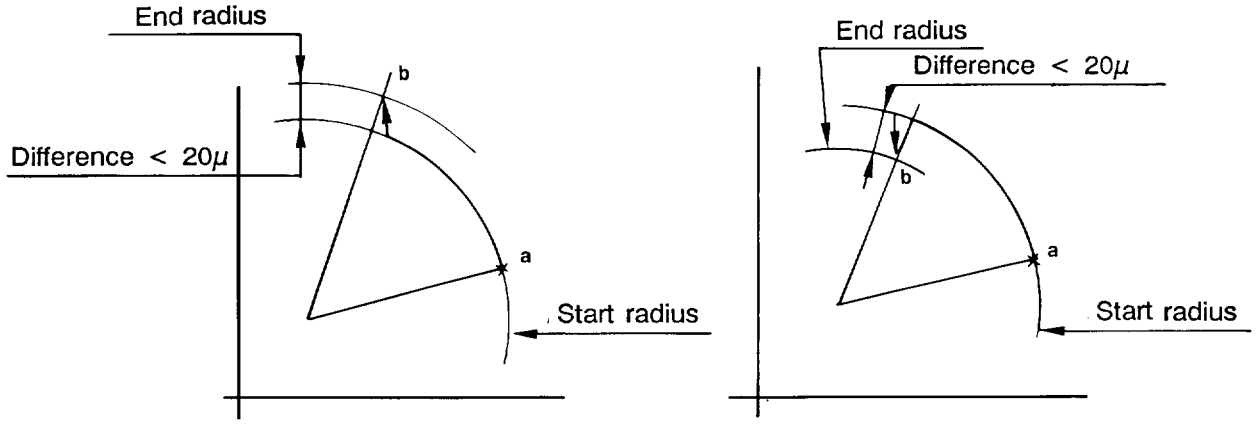
EXAMPLE : Relative programming



N10	G90		X-120.710	Z-220.710			
N20	G2	G91	X-35.355	Z-85.355	I	K-50	F500
N30	G3		X-35.355	Z-85.355	I-35.355	K-35.355	
N40	G1		X-50	Z-50			

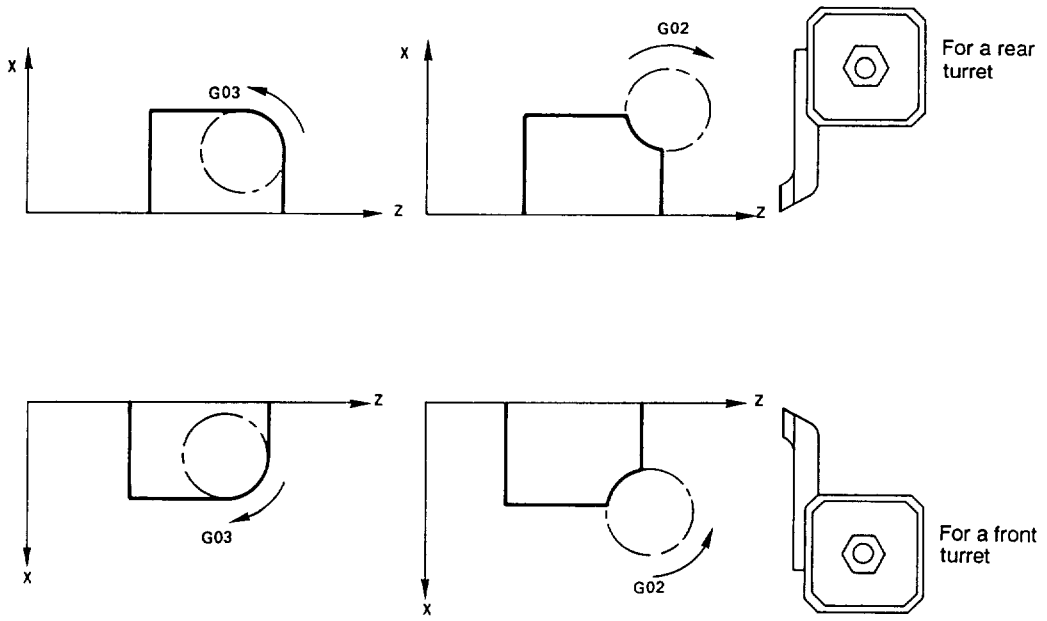
NOTE :

- *If the difference between the start radius and the end radius is greater than 20µ, the system will generate an error message (error 107).*
- *If the radius of the circle is less than 20µ, the trajectory followed will be linear.*

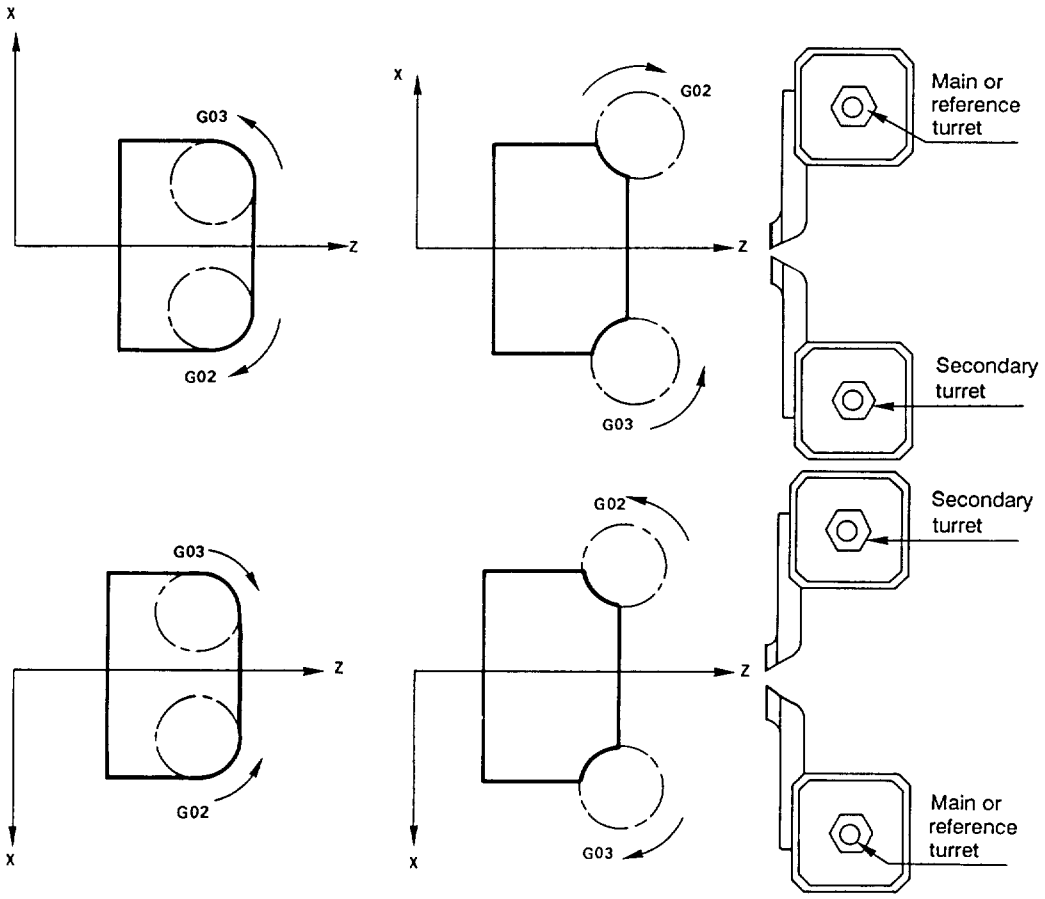


Examples of programming circles on different types of m/c.

- Single turret lathe



- Twin turret lathe



4.1.3 – M19 on spindles fitted with a threading encoder.

By programming M19, the spindle may be orientated to any position relative to the zero position as defined by the m/c manufacturer (see manufacturer's handbook).

The orientation value is given by address C. (C is expressed in 1/1000th of a degree.) The spindle must be running before it can be orientated.

4.2 – PROFILE GEOMETRY PROGRAMMING (PGP)

The NUM 720T allows the workpiece program to be written directly from the drawing dimensions. It calculates inter–connections, and tangential contacts or intersection points which are often left undefined between two path elements on the part drawing. This programming method is only possible in absolute mode (G90).

The classic ISO programming method, (see 4.1), remains valid and can be used in conjunction with GPP.

Programming is carried out in blocks, each block comprising a geometric element (a straight line, or an arc).

A geometric element can be defined completely in a block (eg. the end point of a straight line, or the end point of an arc together with the coordinates of its centre), or the element may be left only partially defined.

If the element is only partially defined, information MUST be provided either in the following block, (or two blocks), which will enable the complete entity, (sequence of blocks), to be defined.

4.2.1 – List of the functions characterizing a geometric element

4.2.1.1 – X..and/or Z.. : coordinates of the end point of a straight line

4.2.1.2 – EA.. : angle of a straight line

4.2.1.3 – X.., Z.. : coordinates of the end point of a circle

4.2.1.4 – I.., K.. : coordinates of the centre of a circle

4.2.1.5 – R.. : radius of a circle

4.2.1.6 – EB+.. : blend radius, or fillet :

The block in which this function is programmed is connected to the following block by a blend radius of the given value.

4.2.1.7 – EB– .. : chamfer :

The block in which this function is programmed is connected to the following block by a chamfer of the given value.

4.2.1.8 – ET : Tangential element

The block in which this function is programmed forms a tangent to the following block.

- Programming of ET is compulsory when it is the only function of the block which characterizes the geometric element : ie. a straight line with known starting point, tangential to the following circle (Fig.10) or a straight line tangential to two circles (Fig.14).
- In all other cases, the programming of ET is optional.

4.2.1.9 – ES : Secant element

The block in which this function is programmed will form a secant, or intersection, with the following block. When the intersection point is not explicitly defined, the ES function MUST be programmed in the first of the two blocks.

4.2.1.10 – E{+/−}: Discriminant

The discriminant E+ or E– is used to choose between two mathematically possible solutions.

- The discriminant may be combined with ET and ES :
  Example :
  ES– is equivalent to ES E–
  ET+ is equivalent to ET E+

For a line–circle or a circle–circle intersection, there are only two possible solutions and the discriminant MUST be programmed.

For tangential elements, there are several possible solutions. To limit these, the system executes only "continuous" tangents (ie. not turning back,) thus reducing the maximum number of solutions to two.

When two solutions are possible, one involves the creation of an arc of less than 180° and the other, an arc greater than 180°. Programming the discriminant is optional. By default, the system will choose the solution which gives the smaller arc.

The only exception is a circle whose centre lies inside the following circle, and which is characterized only by the coordinates of this centre and by the fact that it is tangential to the following circle (Fig. 5b, 12b, 23b). See paragraph 4.2.3.2.

4.2.2 – Entity programming – Choosing the discriminant

The group of blocks required by the system to calculate the coordinates of a geometric element (end point and/or centre of the circle) constitute a "geometric entity" originating from the starting point of its first block. This point can be :

- either fully defined by the preceding block,

- or already calculated by the system, given that the first block of a geometric entity may be the last block of the preceding entity.

Where a discriminant is used to define an element of a geometric entity, it must be programmed in the first block of this entity. The + and – signs will then specify the position of the characteristic point (intersection, tangent, circle centre) of the desired solution, with respect to a constructed line (D).

NOTE :

The constructed straight line (D) is :

- *the straight line, defined by its angle EA..., (if one of the elements of the entity is defined in this way,)*

- *or the straight line connecting a known point in the first element to a known point in the last element of the entity (pointing from the first towards the last). This known point is normally the centre of a circle programmed by I..K.., or, another given point. (The start point of the first element or the end of the last.)*

Having constructed the straight line (D), there are two possibilities :

- *either the points characterizing the two possible solutions are situated along the straight line (D), and*
  - *E+ then defines the point closest to + ∞ on the straight line, and*
  - *E– defines the point closest to – ∞ on the straight line.*

- *or, the points characterizing the two possible solutions are on either side of the straight line (D), and*
  - *E+ then defines the point to the left of (D), and*
  - *E– defines the point to the right of (D)*

4.2.3 – Block format authorized in programming a profile

Symbols used in the following examples :

{ . . } { . . } { . . }	Choice of several functions or items of data
[. .]	Optional function
/	Contact symbol
≠	Intersection symbol

4.2.3.1 – A geometric element completely defined in one block

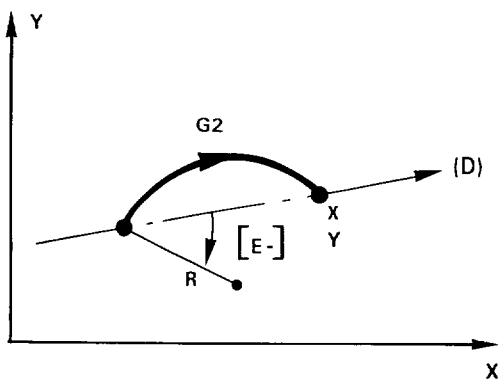
4.2.3.1.1 – Straight line

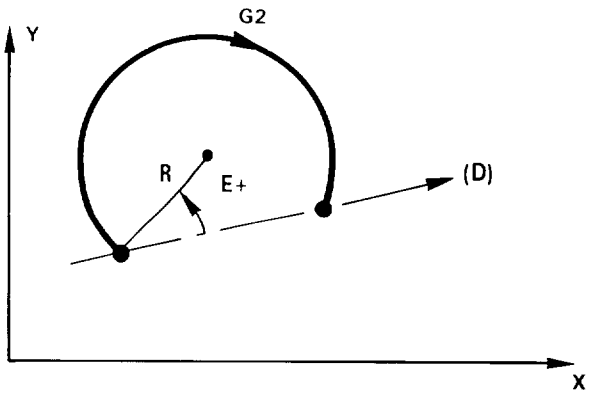
G01 { X.. / Z.. / X.. Z.. / EA.. X.. / EA.. Z.. }

4.2.3.1.2 – Circle

G {2 / 3} X.. Z.. I.. K..

G {2 / 3} X.. Z.. R.. (E ±)





4.2.3.2 – A geometric element defined over more than one block.

- The first block is a straight line,
- Whose start point is completely defined.

1st BLOCK	2nd BLOCK	3rd BLOCK	TYPE OF PROFILE	FIGURES
G1 EA.. ES	EA.. X.. Y		line ≠ line	FIG. 1
	EA..	G {2 3} I.. J.. {R.. X.. Y..}	line ≠ line/circ	FIG. 2
G1 EA.. ES {+ -}	G {2 3} I.. J.. {R.. X.. Y..}		line ≠ circ	FIG. 3
	G {2 3} I.. J..	G1 EA.. X.. Y..	line ≠ circ/line	FIG. 4
	G {2 3} I.. J.. [ET {+ -}]	G {2 3} I.. J.. {R X.. Y..}	line ≠ circ/circ	FIG. 5
G1 EA.. [ET]	G {2 3} I.. J..		line/circ	FIG. 6
	G {2 3} R..	G1 EA.. X.. Y..	line/circ/line	FIG. 7
G1 EA.. [ET {+ -}]	G {2 3} R.. X.. Y..		line/circ	FIG. 8
	G {2 3} R..	G {2 3} I.. J.. {R.. X.. Y..}	line/circ/circ	FIG. 9
G1 ET	G {2 3} I.. J.. {R.. X.. Y..}		line/circ	FIG. 10
	G {2 3} I.. J..	G1 EA.. X.. Y..	line/circ/line	FIG. 11
	G {2 3} I.. J.. [ET {+ -}]	G {2 3} I.. J.. {R.. X.. Y..}	line/circ/circ	FIG. 12

- The first block is a circle,
- Whose start point is completely defined.

1st BLOCK	2nd BLOCK	3rd BLOCK	TYPE OF PROFILE	FIGURES
G{2/3} I.. J.. [ET]	G1 { EA.. / X.. / Y.. / EA.. X.. / EA.. Y.. / X.. Y.. }		circ/line	FIG. 13
	G1 ET	G{2/3} I.. J.. { R.. / X.. Y.. }	circ/line/circ	FIG. 14
G{2/3} I.. J.. [ET {+/-}]	G{2/3} I.. J..		circ/circ	FIG. 15
	G{2/3} R.. X.. Y..		circ/circ	FIG. 16
	G{2/3} R..	G1 EA.. X.. Y..	circ/circ/line	FIG. 17
		G{2/3} I.. J.. { R.. / X.. Y.. }	circ/circ/circ	FIG. 18
G{2/3} I.. J.. ES {+/-}	G1 EA.. X.. Y..		circ≠line	FI.G 19
	G1 EA..	G{2/3} I.. J.. { R.. / X.. Y.. }	circ≠line/circ	FIG. 20
	G{2/3} I.. J.. { R.. / X.. Y.. }		circ≠circ	FIG. 21
	G{2/3} I.. J..	G1 EA.. X.. Y..	circ≠circ/line	FIG. 22
	G{2/3} I.. J.. [ET {+/-}]	G{2/3} I.. J.. { R.. / X.. Y.. }	circ≠circ/circ	FIG. 23
G{2/3} R.. [ET {+/-}]	G1 EA.. X.. Y..		circ/line	FIG. 24
	G1 EA..	G{2/3} I.. J.. { R.. / X.. Y.. }	circ/line/circ	FIG. 25
	G{2/3} I.. J.. { R.. / X.. Y.. }		circ/circ	FIG. 26
	G{2/3} I.. J..	G1 EA.. X.. Y..	circ/circ/line	FIG. 27
	G{2/3} I.. J.. [ET {+/-}]	G{2/3} I.. J.. { R.. / X.. Y.. }	circ/circ/circ	FIG. 28

FIG. 1

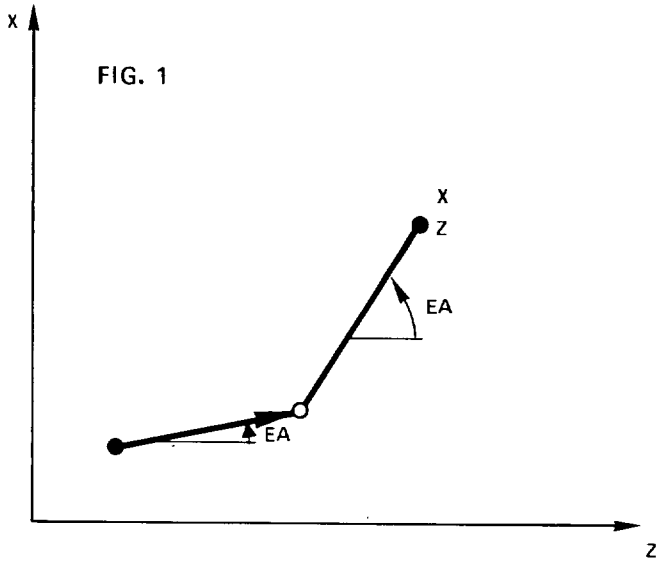


FIG. 2

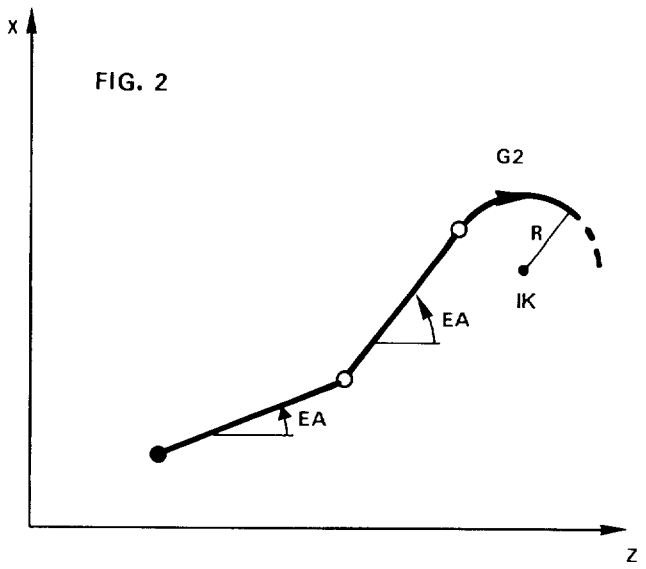


FIG. 3a

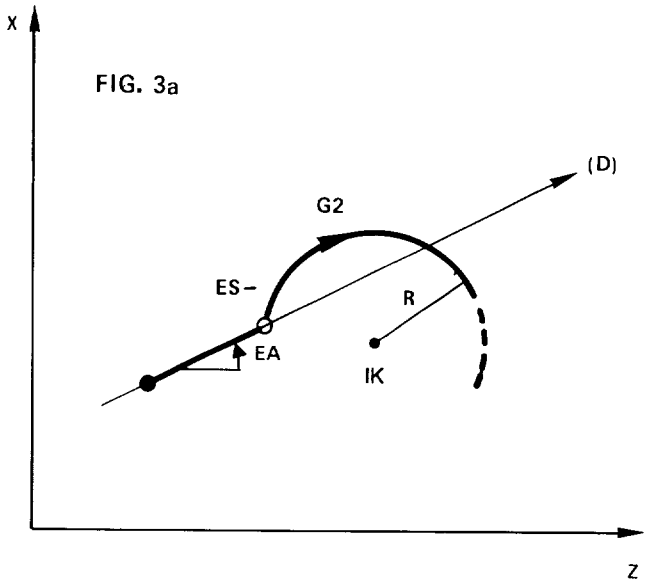


FIG. 3b

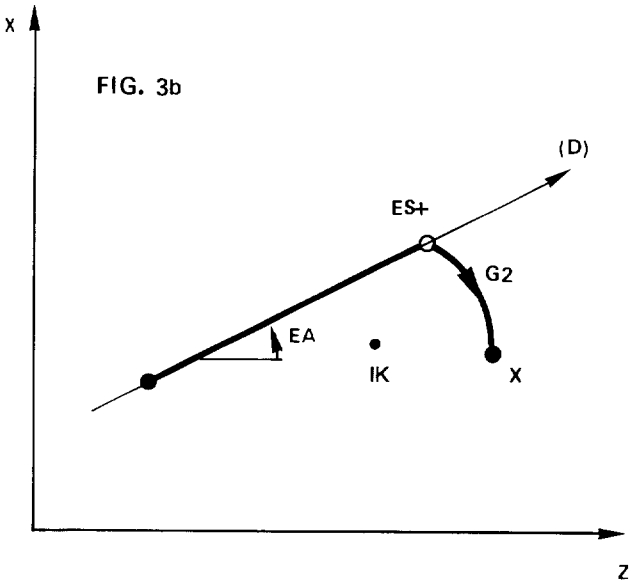


FIG. 4a

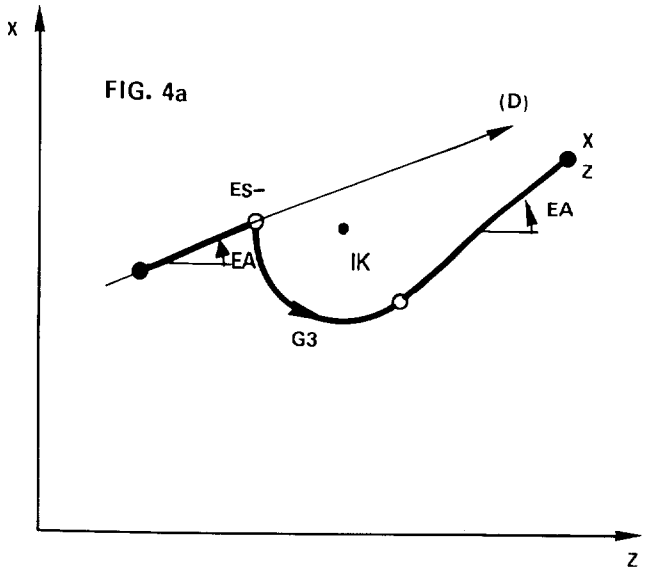


FIG. 4b

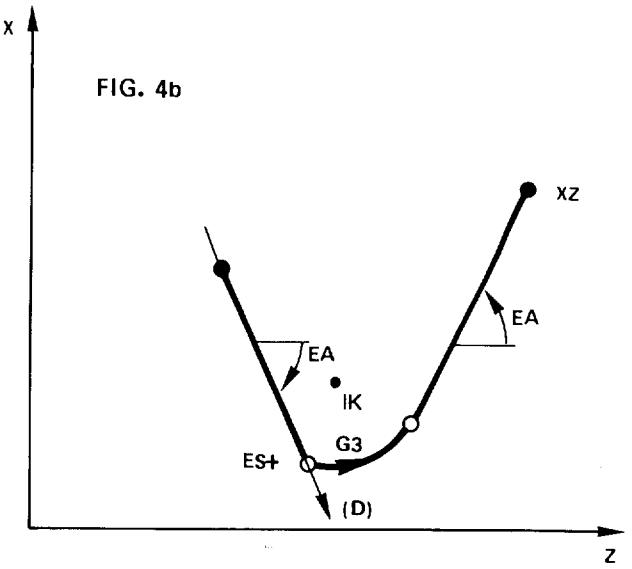


FIG. 5a

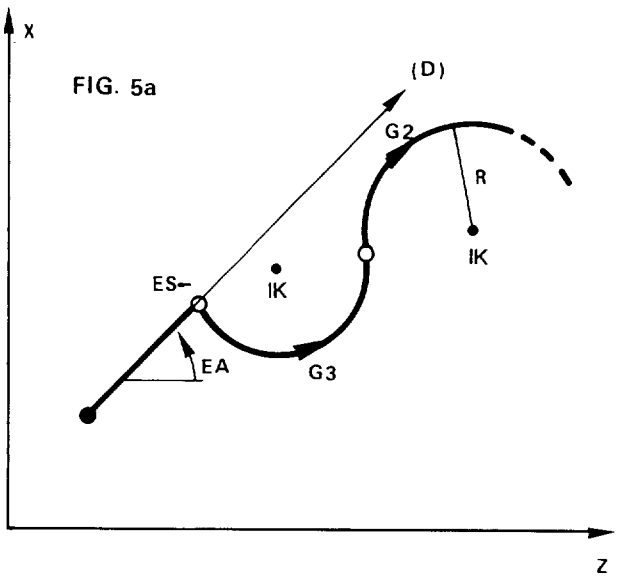


FIG. 5b

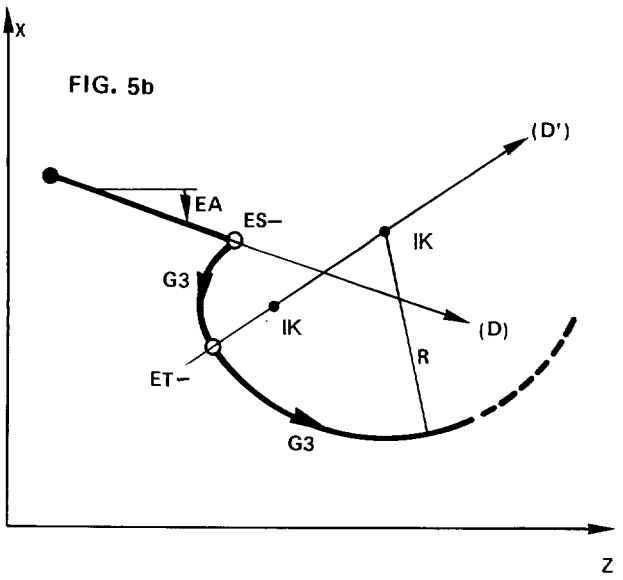


FIG. 6

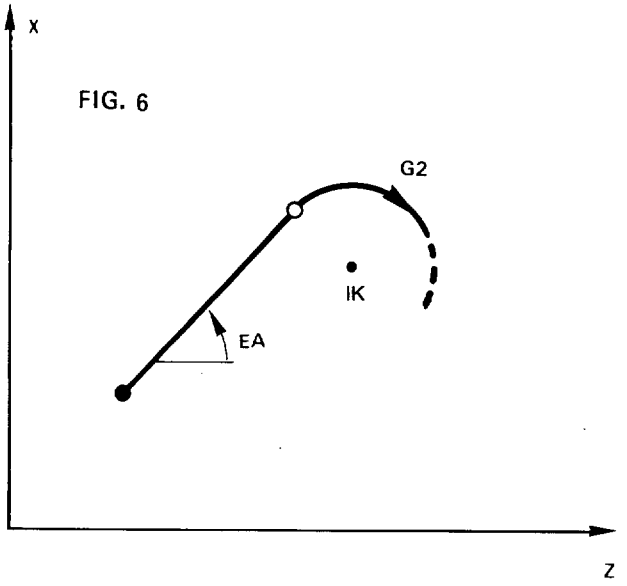


FIG. 7

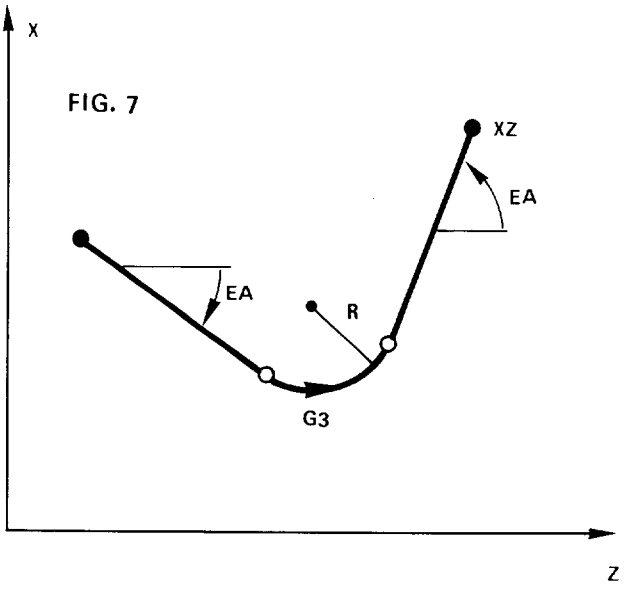


FIG. 8a

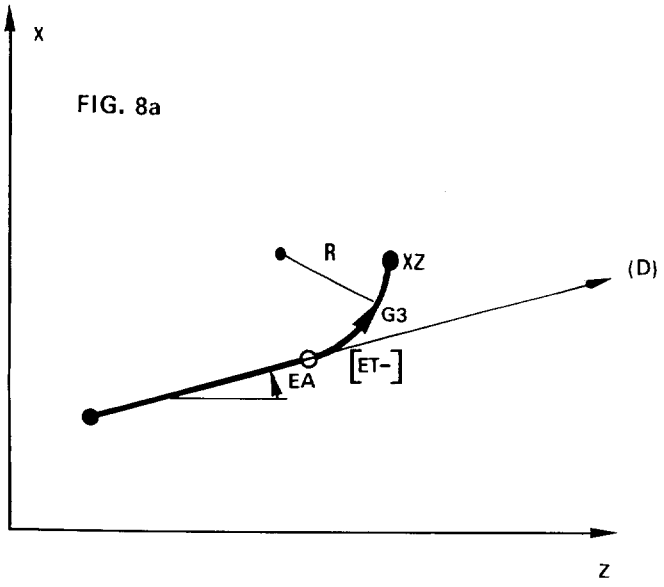


FIG. 8b

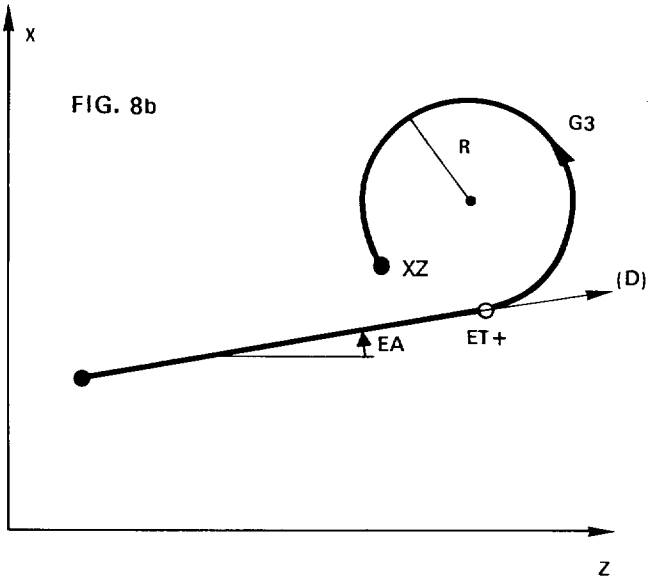


FIG. 9a

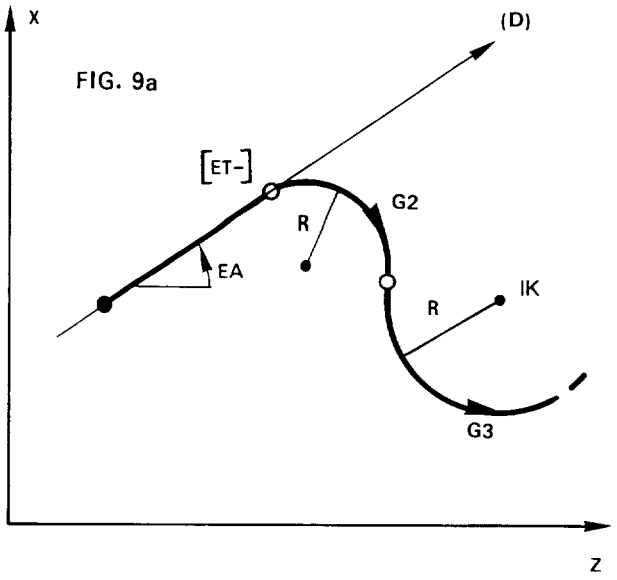


FIG. 9b

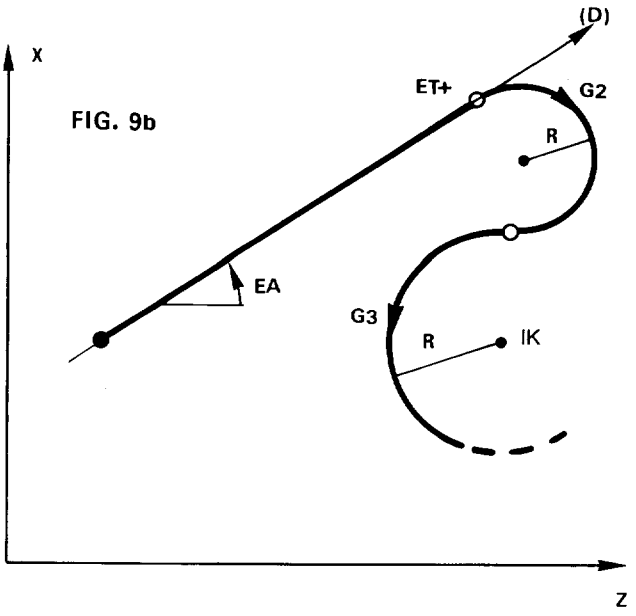


FIG. 10

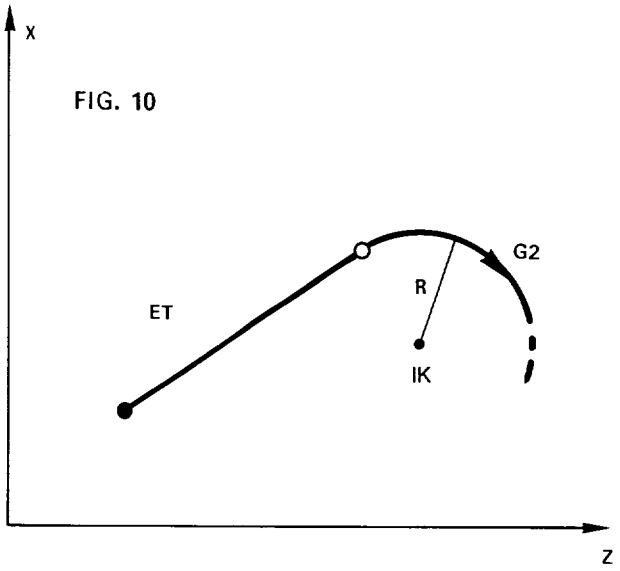


FIG. 11

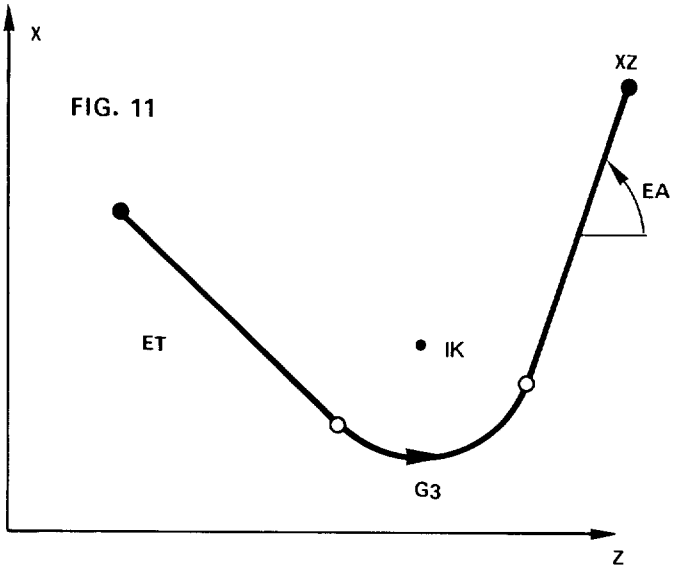


FIG. 12a

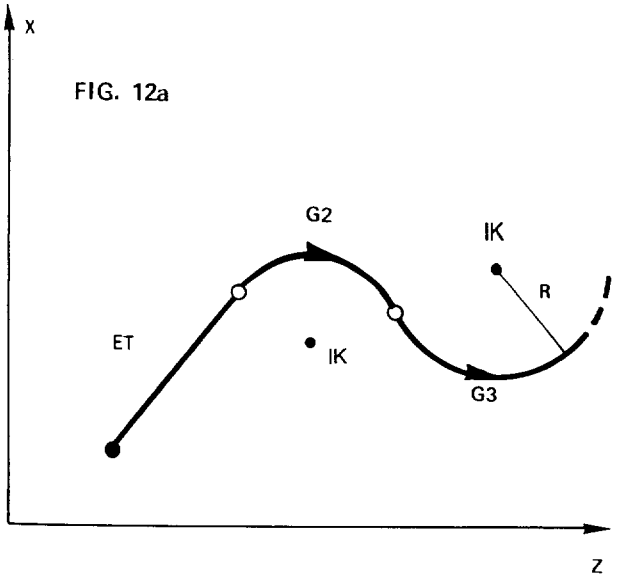


FIG. 12b

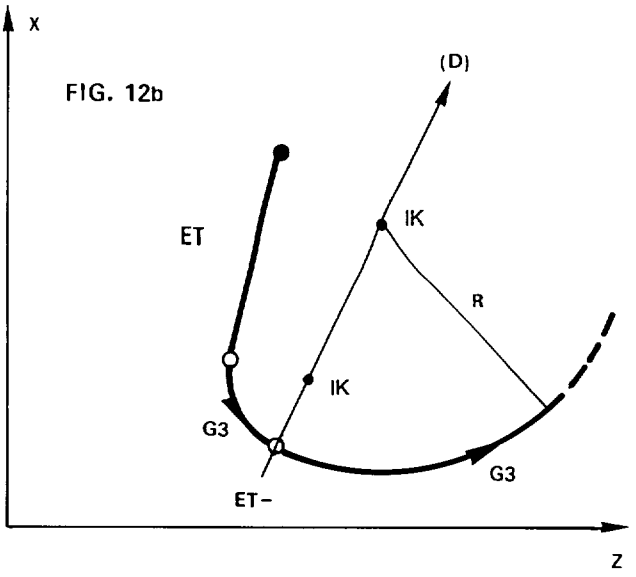


FIG. 13a

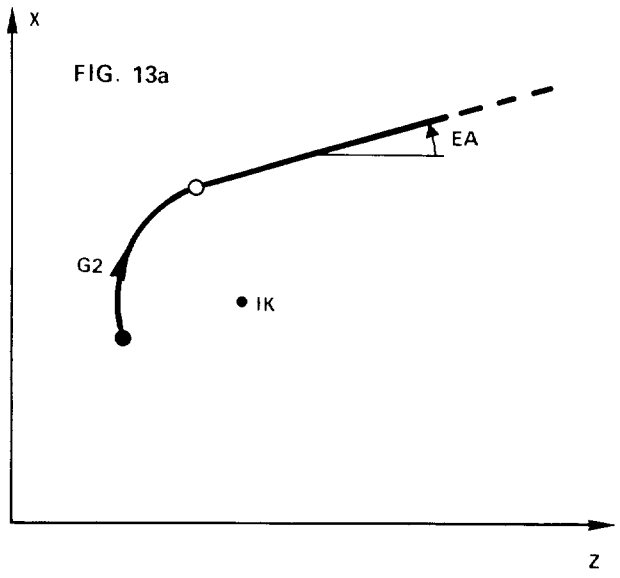


FIG. 13b

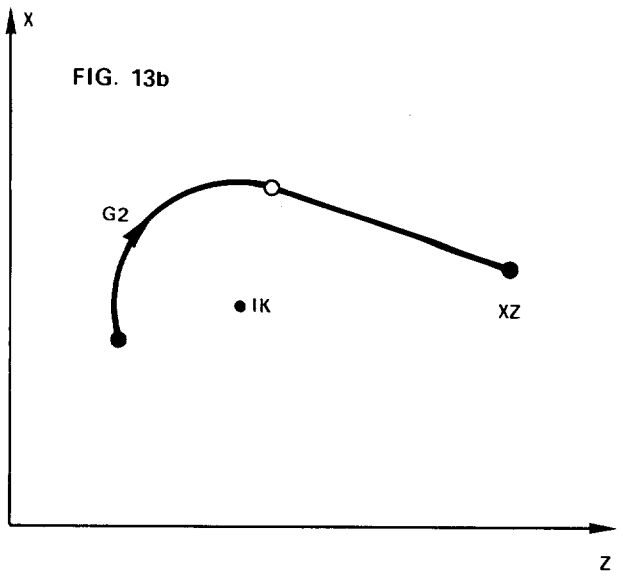


FIG. 14a

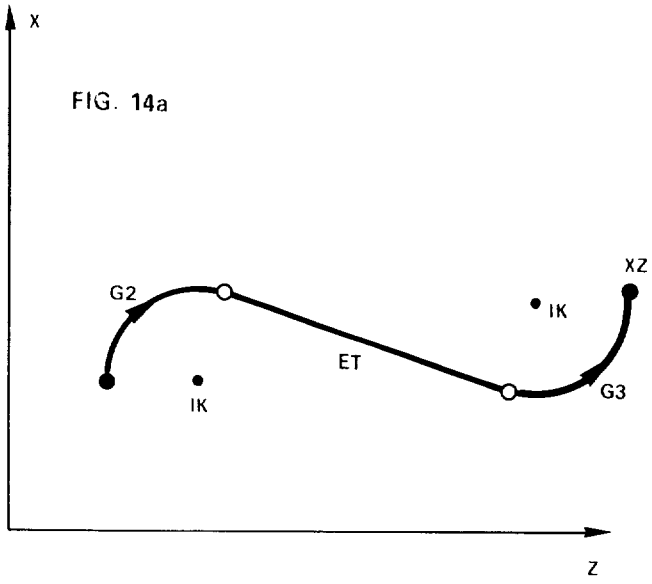


FIG. 14b

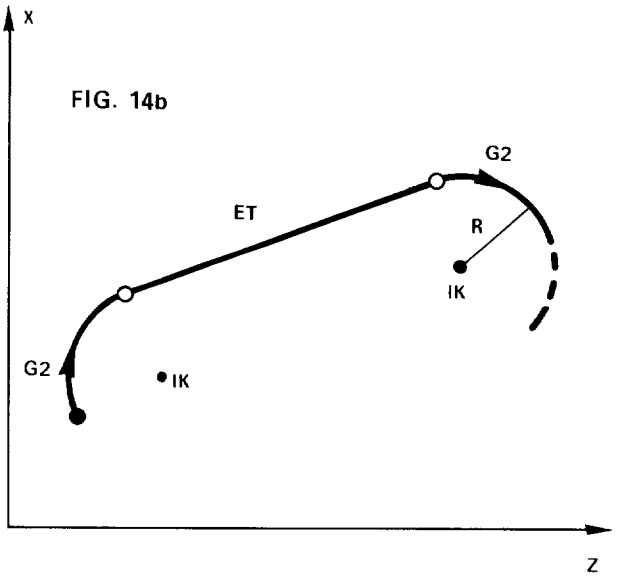


FIG. 15a

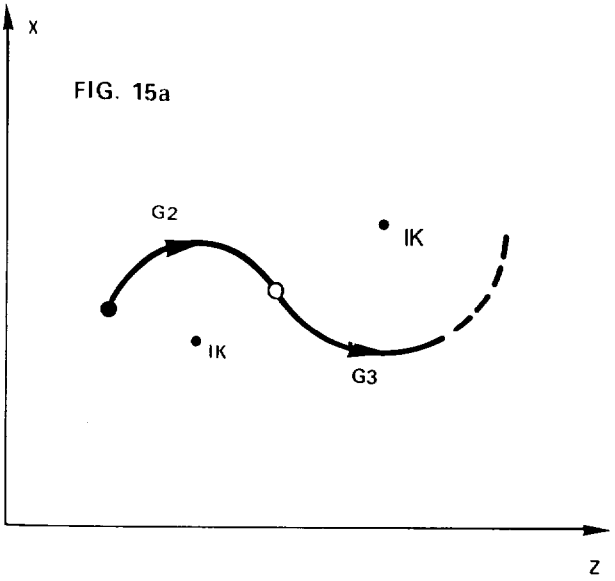


FIG. 15b

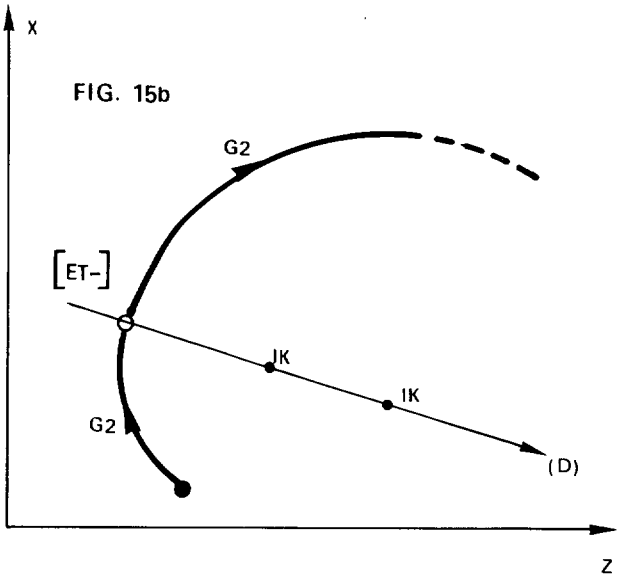


FIG. 16

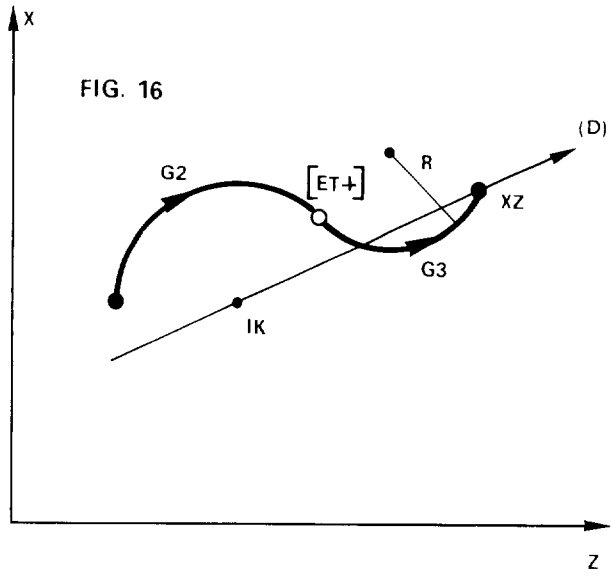


FIG. 17

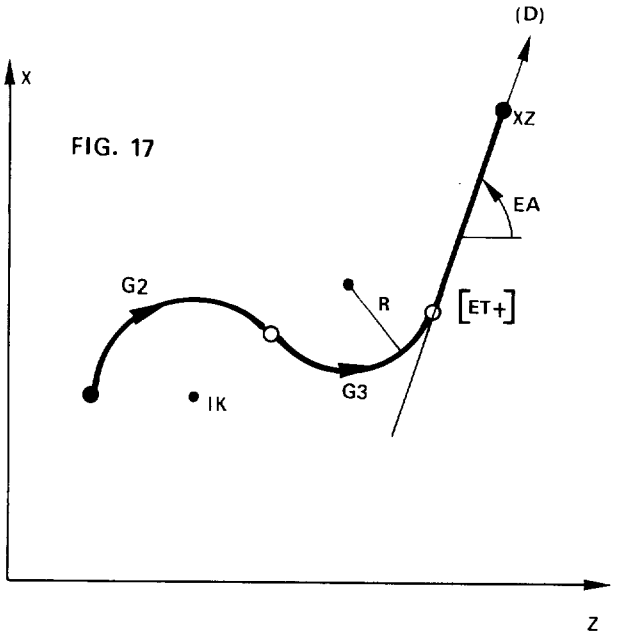


FIG. 18a

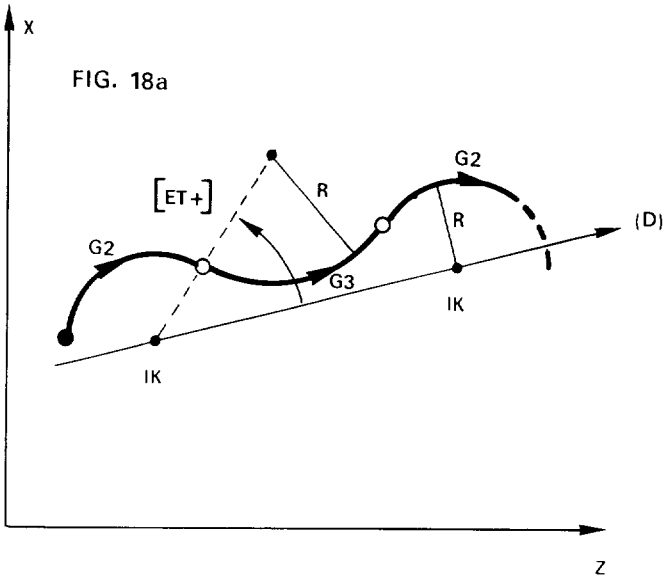


FIG. 18b

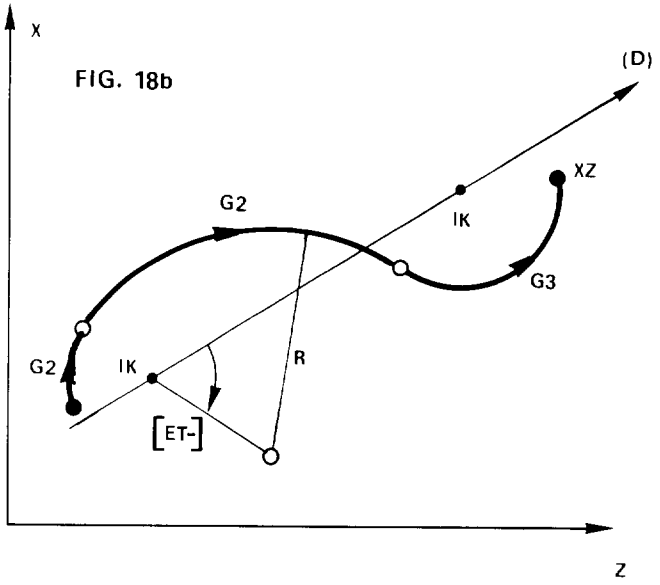


FIG. 19

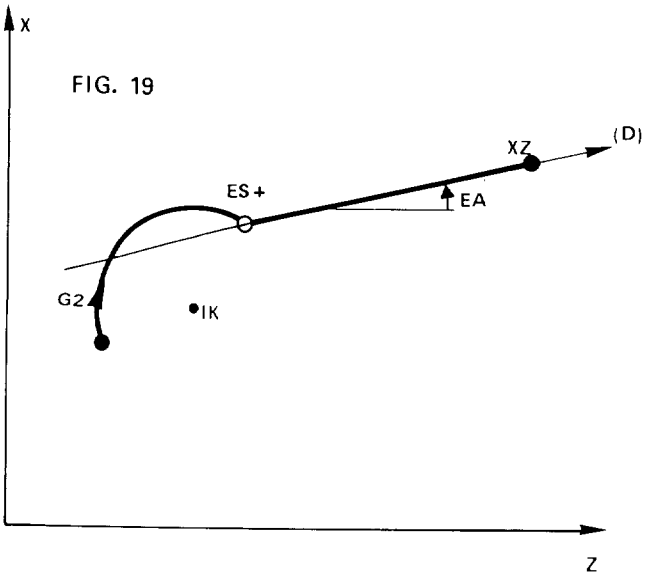


FIG. 20

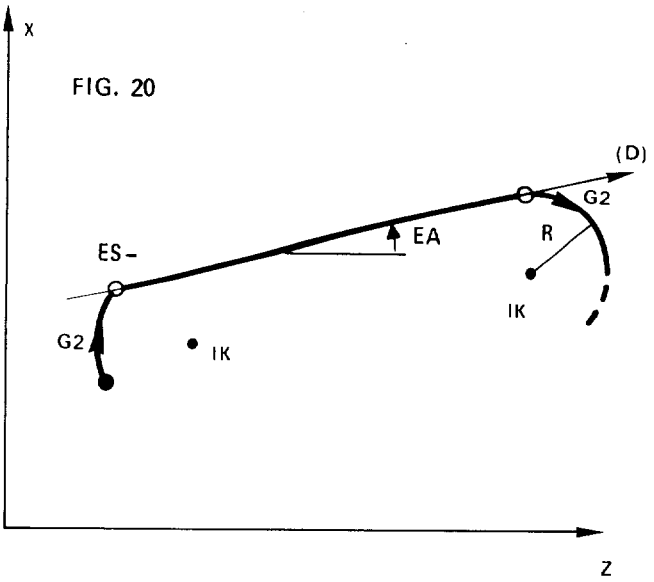


FIG. 21

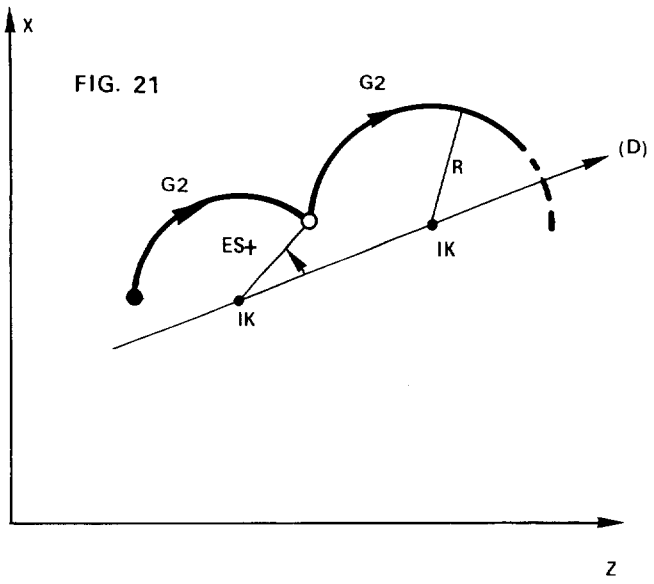


FIG. 22

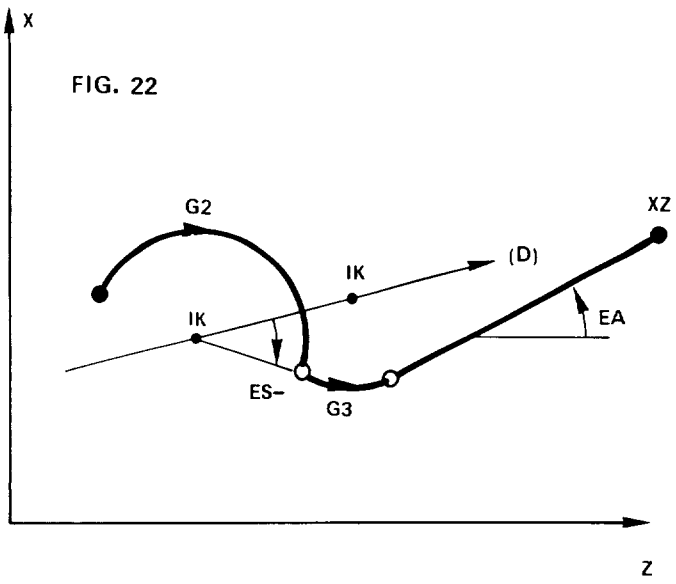


FIG. 23a

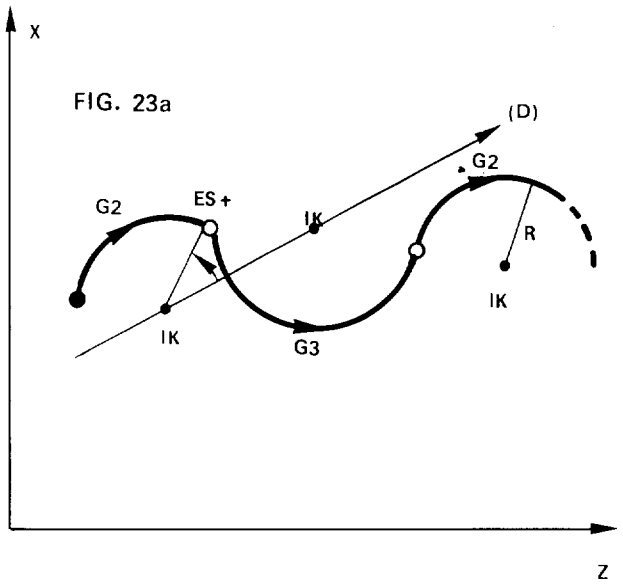


FIG. 23b

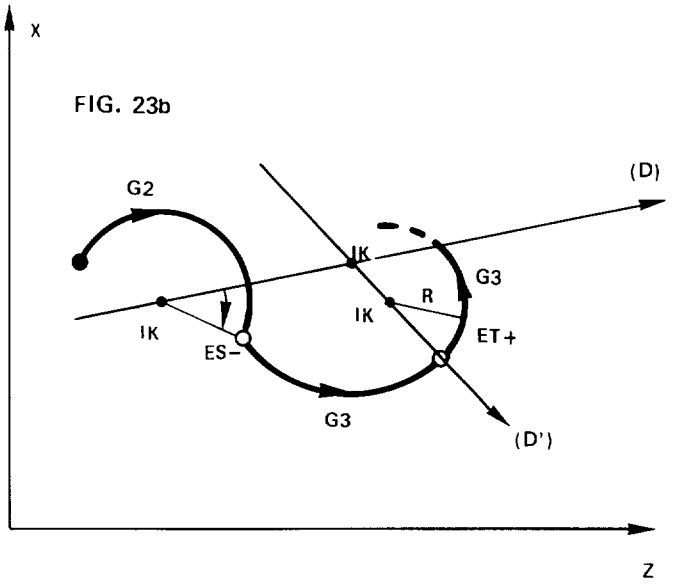


FIG. 24a

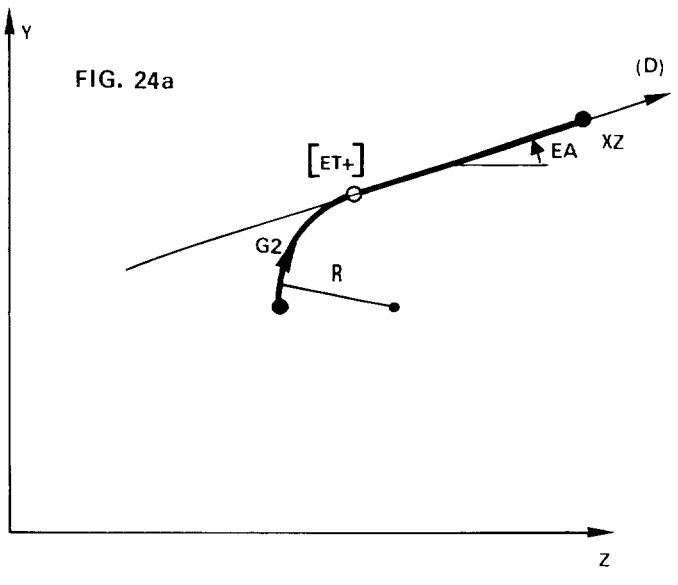


FIG. 24b

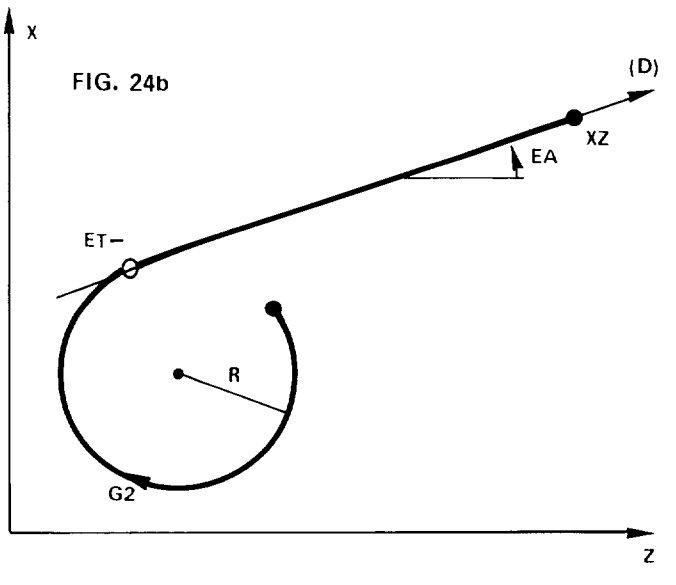


FIG. 25

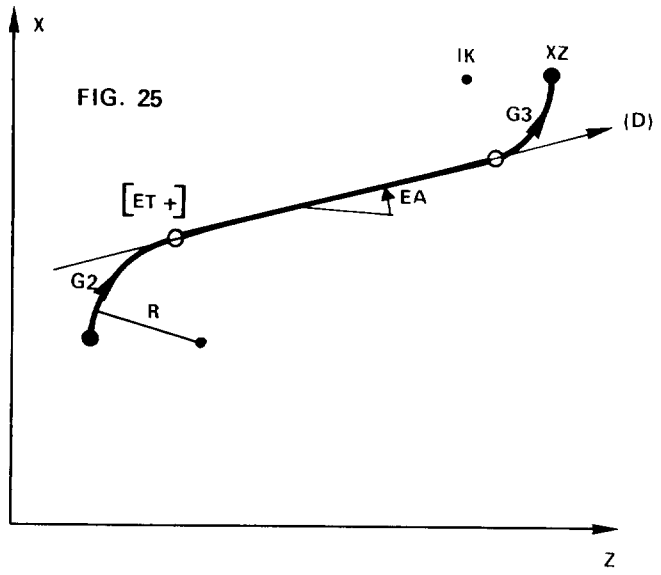


FIG. 26

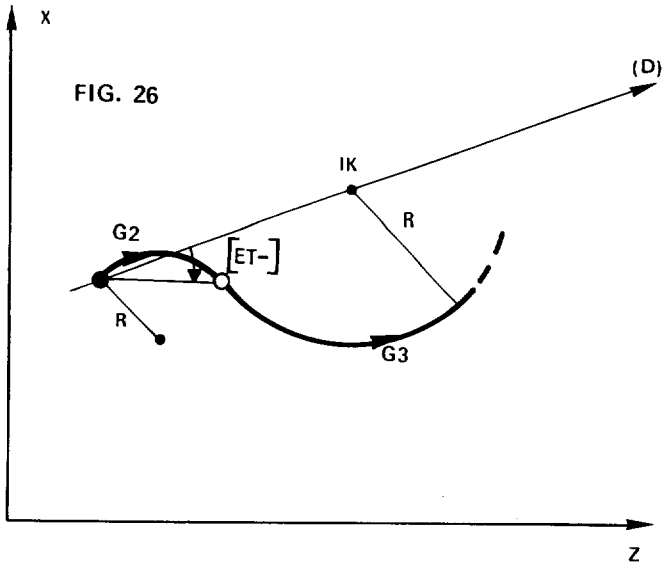


FIG. 27a

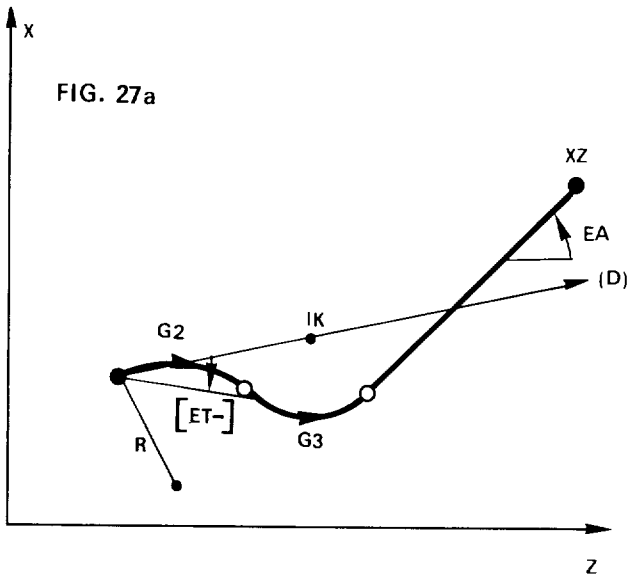


FIG. 27b

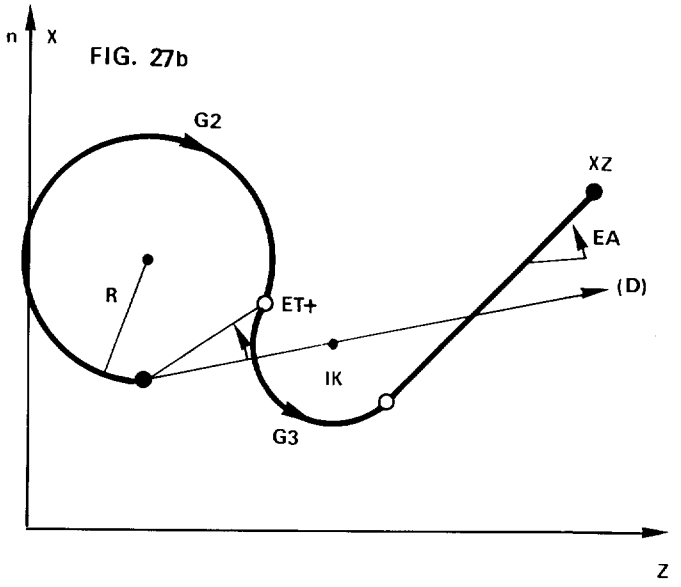


FIG. 28a

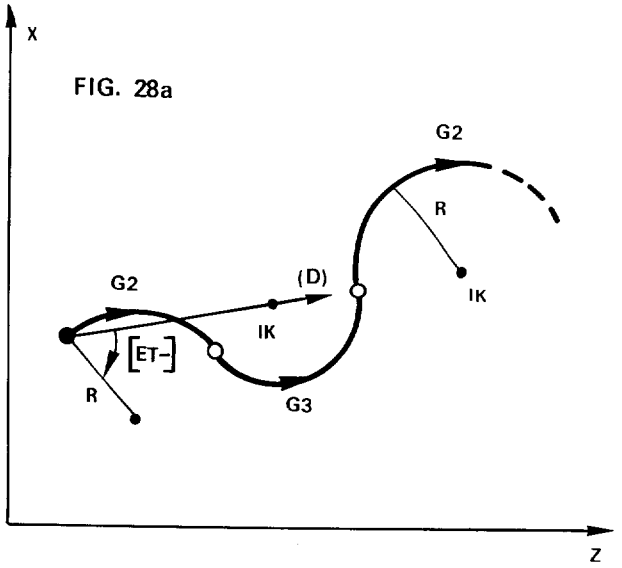
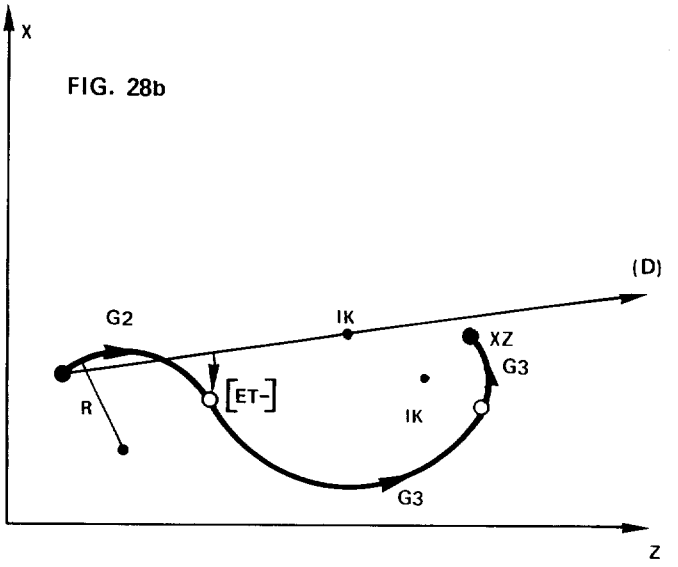


FIG. 28b

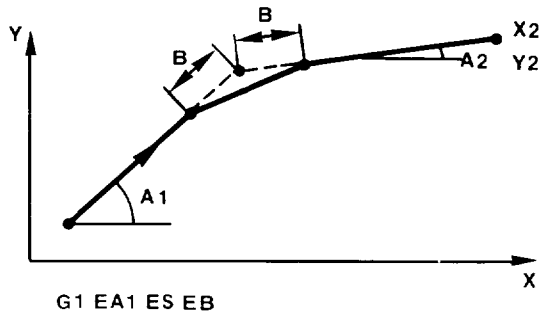


4.2.3.3 – Programming chamfers and blends

A chamfer can only be inserted between two successive straight lines. A blend radius may be inserted between any two elements, lines or circles.

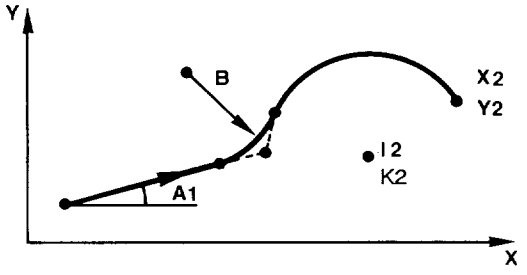
A chamfer is programmed by the code EB– and the chamfer value. A blend is programmed by the code EB+ and the value of the radius.

Chamfer



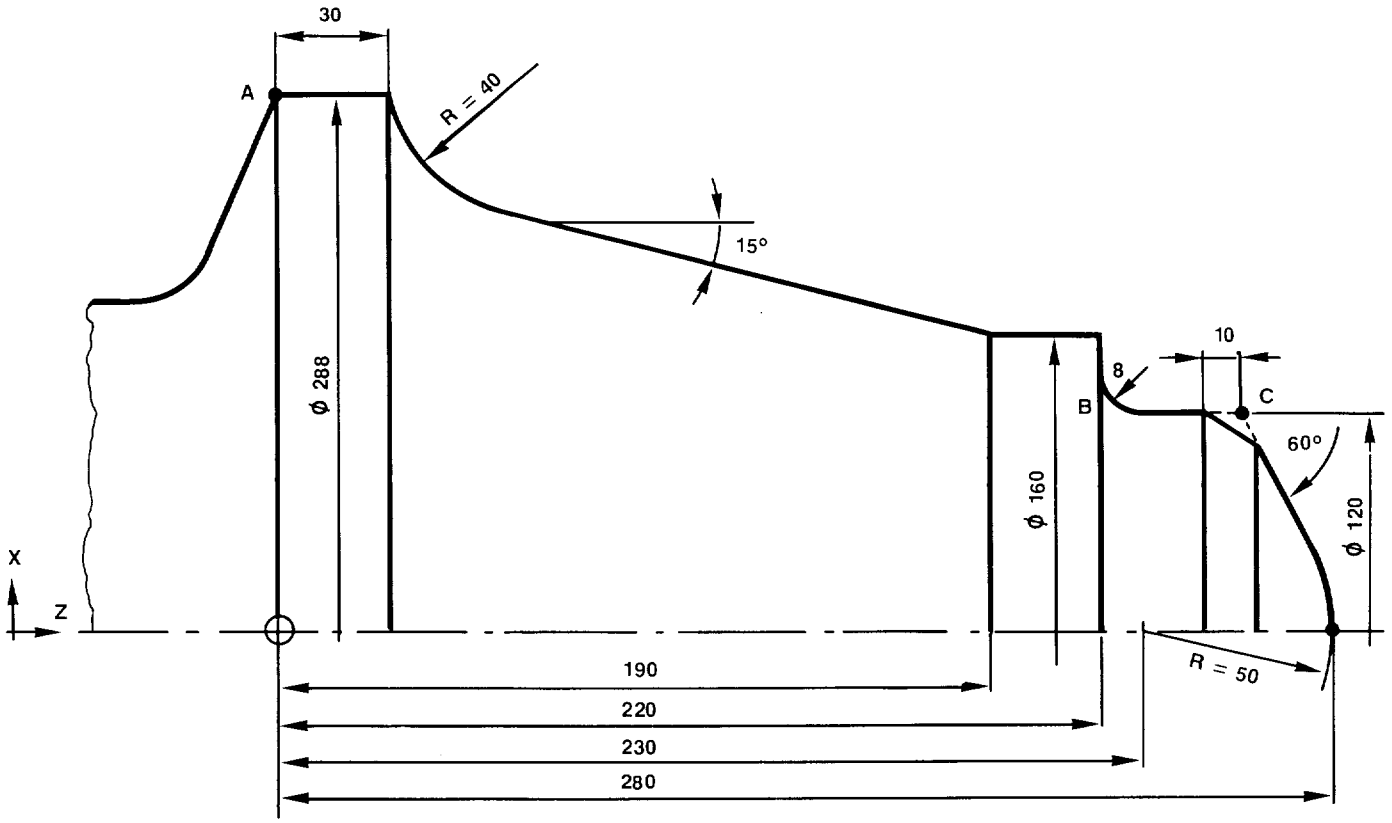
G1 EA1 ES EB
G1 EA2 X2 Z2

Blend

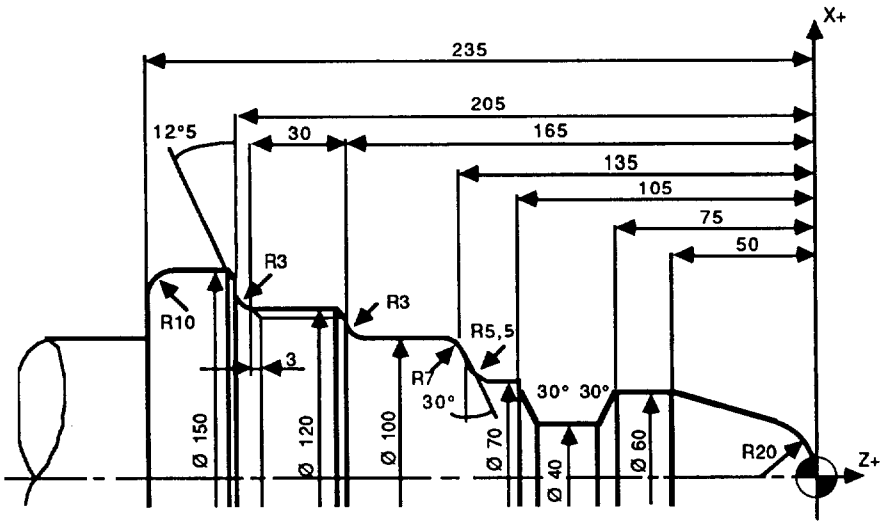


G1 EA1 ES EB+
G2 K2 J2 X2 Z2

4.3 – PROGRAMMING EXAMPLES



N100 X288 Z0 (point A)
N110 G1 Z30
N120 G3 R40
N130 G1EA–15 X160 Z190
N140 Z220
N150 X120 EB8 (blend radius at B)
N160 EA ES EB–10 (chamfer at C)
N170 EA–60
N180 G2 X0 Z280 I0 K230



N10 X0 Z0
N20 G3 I0 K-20
N30 G1 X60 Z-50
N40 Z-75
N50 EA-120 X40
N60 EA180 ES
N70 EA120 X60 Z-105
N80 X70
N90 EA180 EB5.5
N100 EA120 X100 Z-135 EB7
N110 Z-165 EB3
N120 X120 EB-2
N130 EA180 EB3 ES
N140 EA102.5 X150 Z-205 EB-2
N150 Z-233

5 - TOOL PROGRAMMING

The tool number is programmed with the T address (3 digits). Its compensation number is specified with the D address (2 digits).

Any compensator can be assigned to any tool.

5.1 - PROGRAMMING THE TOOL NUMBER

T... the tool number is modal. (Max. value, 255)
This code is normally used in conjunction with M6, to carry out a manual or automatic tool change.

5.2 - TOOL COMPENSATIONS

The system has 32 compensators (each with X, Z, R, and C values) which can be used to accomodate tool and tip radius dimensions.

5.2.1 - Enabling compensation

Programming D.. ENABLES the associated compensations. No axis movements are generated by changing the D compensation. The compensations will ONLY be taken into account during the next programmed move of the axis.

Maximum values that can be entered through the keypad :

X $\pm$ 9999.999 mm

Z $\pm$ 9999.999 mm

R + 999.999 mm

C 0 to 8

5.2.2 - Cancelling compensation

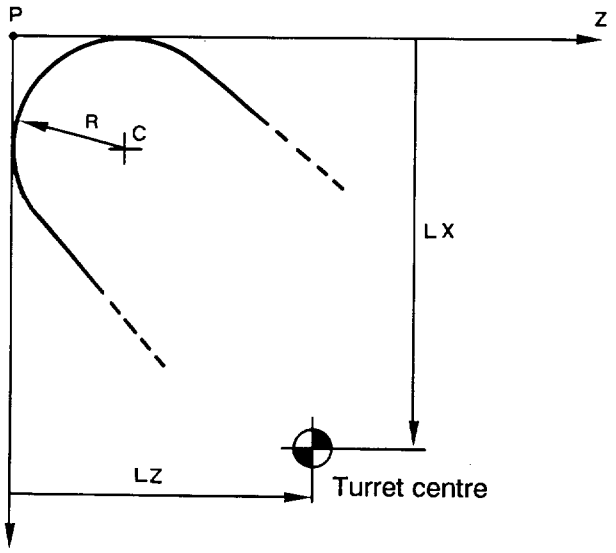
All tool compensations are cancelled by programming compensator D0, whose contents are always zero. (Compensations are not active in G52 blocks.)

Address D and its value is modal.

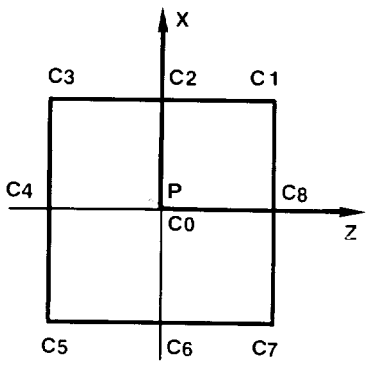
5.3 – TOOL NOSE ORIENTATION (C)

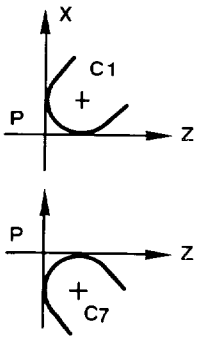
Tool dimensions are measured upto the cutting edges of the tools parallel to the X and Z axes. To cut a profile, the tool radius must be known, in order to keep the compensation normal to the programmed path.

If the theoretical cutting point is P, then when tool radius compensation is programmed (G41 or G42), the system carries out a translation along vector PC and then calculates and applies a compensation vector to keep the radius R normal to the profile. The vector, PC, is set for each tool by a code, (from C0 to C8,) entered via the keypad.

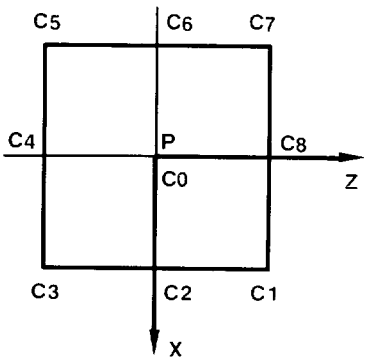


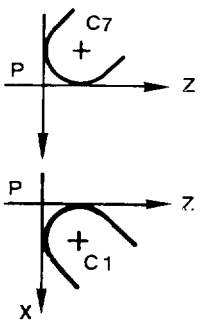
EXAMPLES





REAR TURRET





FRONT TURRET

5.4 – TOOL DIMENSIONS

There are three possible ways of entering tool dimensions into the memory :

- Manual input via the keypad (the TOOL key)

  Format : D2 X± .... Z± .... R+ .... C0 to 8

  D2 indicates the compensation number.

  The dimensions may be entered when the machine is idle or busy.

- Semi–automatic input : TSET mode

  Tool values are calculated directly by the CNC after stopping at a setting piece whose dimensions have been pre–stored in the memory.

  The data that should be input is :

  D2 X R + 3.3 C1* X or Z without dimensions
  or D2 X R + 3.3 C1*

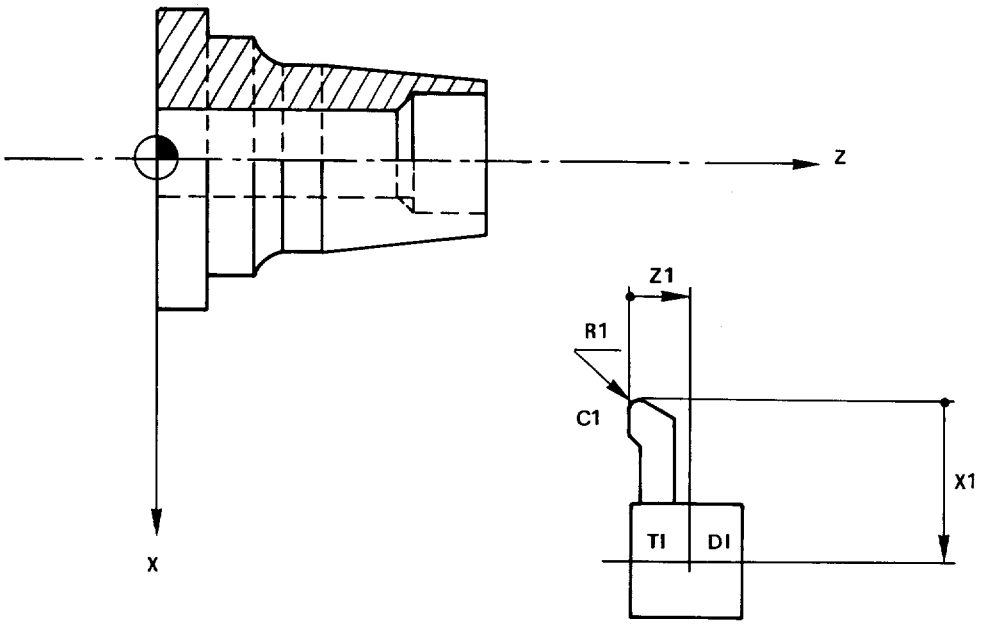
- Input by tooling tape or optional link (DNC1)

  The "data load" mode is mutually exclusive of other modes (such as machining mode) and is obtained by pressing the load key.

  The format of the data is :

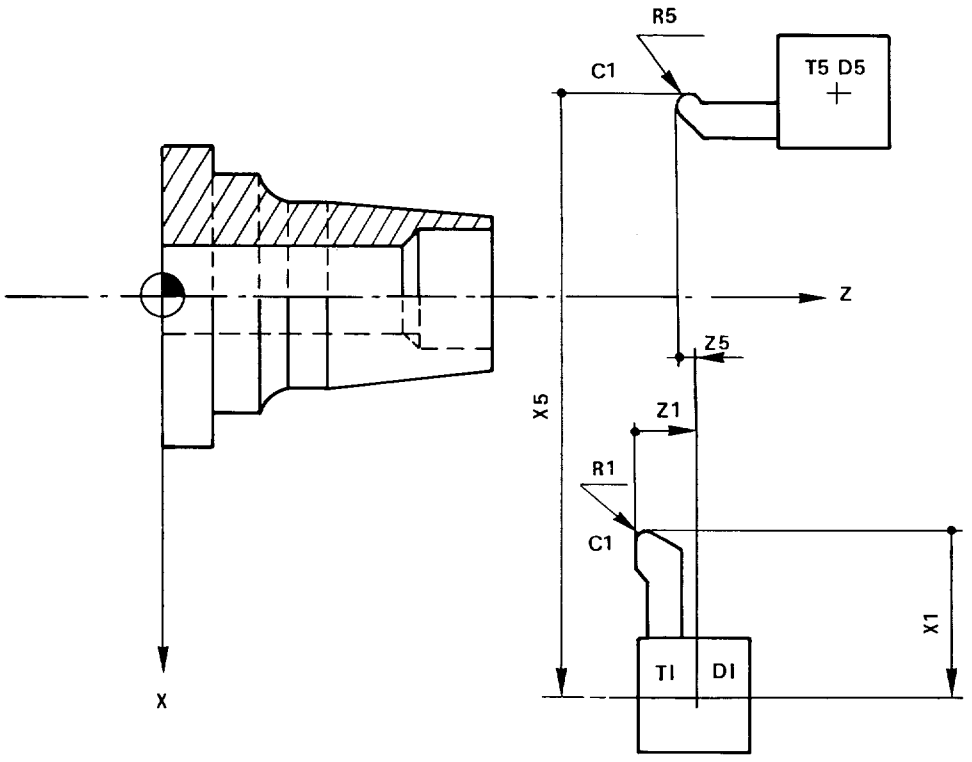
  % D2 X±4.3 Z±4.3 R+3.3 C1
  '
  '
  '
  '
  Dn ---------- X OFF

Single turret (front)



Twin turrets

When there are two turrets, the tools of the secondary turret are defined with respect to the reference point of the main turret (the front turret in this example).



5.5 – TOOL WEAR COMPENSATIONS

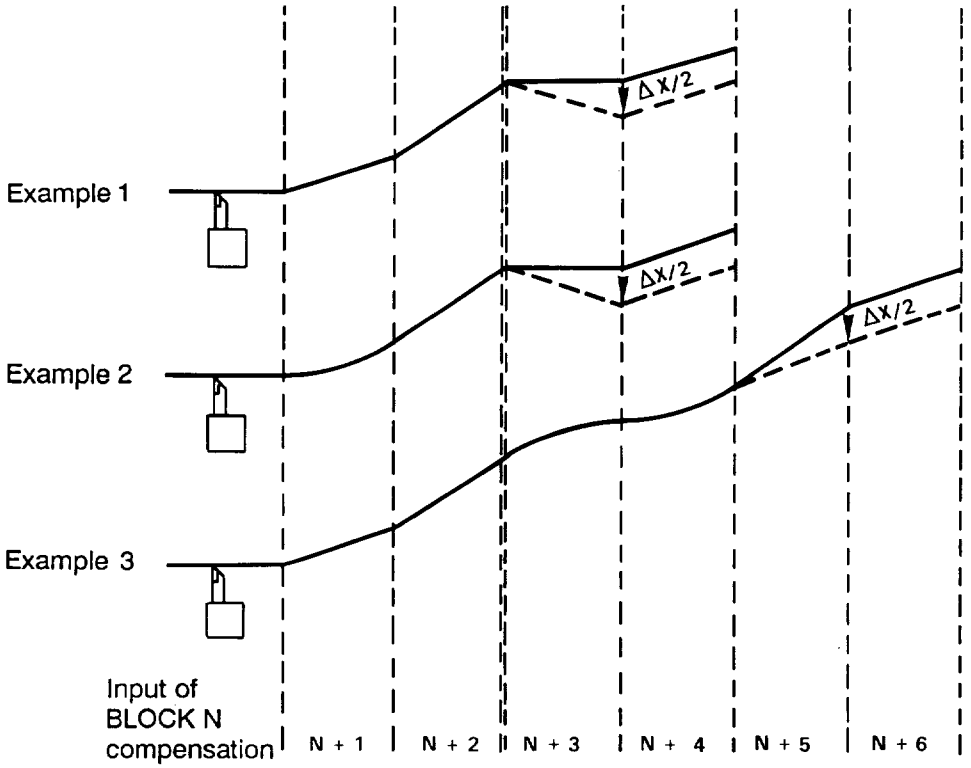
Tool dimensions can be modified, within a limited range, by wear compensation values. Wear values are accumulated separately for each compensator and can be compared with a threshold, using parameterized programming techniques, from the part program or a macro.

The maximum value which can be manually input in one go, is ±0.999mm.

Wear compensations in the X axis are always input on diameter, for ease of entry reasons.

The running total of each wear compensator can be zeroed, by the operator. (It is also reset to zero by changing the respective axis tool dimension.)

Wear compensation values input at block N, during machining, take effect as from the first linear interpolation block which follows block N+2.



5.6 – USING TOOL DIMENSIONS AND WEAR COMPENSATIONS

Tool compensations are activated by the D address and are independent of the tool number, which is designated by the T address.

The maximum storage capacity is 32 compensators. Each compensator is composed of the tool dimensions in the X and Z axes, the tool nose radius, the tool nose centre orientation and the wear compensations in the X and Z axes.

Maximum values allowed are :

LX or LZ ± 9999.999 mm : tool dimensions

ΔX or ΔZ ± 999.999 mm : wear values

R + 999.999 mm : tool nose radius

Cn : tool nose orientation. (n is from 0 to 8)

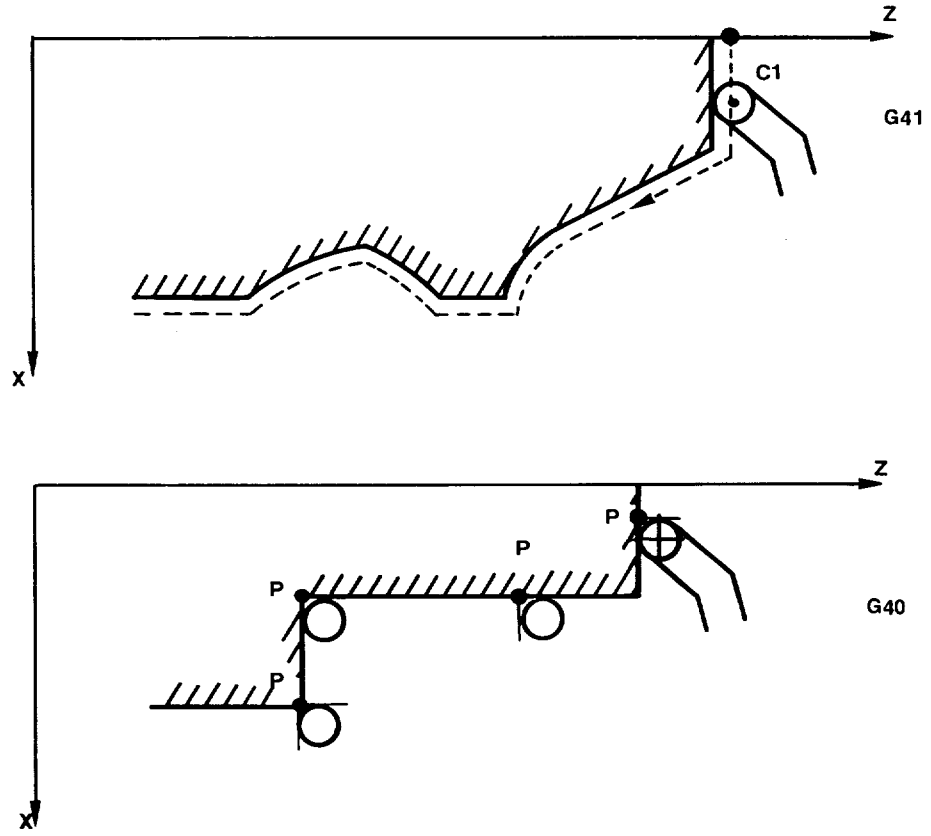
Tables of tool dimensions and tool wear are both displayed on the TOOL page. Press the TOOL key repeatedly to alternate between the two tables.

The table of tool dimensions may be transferred to a tape or other peripheral by pressing the UNLOAD key.

The operator can enter both the compensator number and the compensation values via the keypad.

5.7 – TOOL RADIUS COMPENSATION – G41/G42

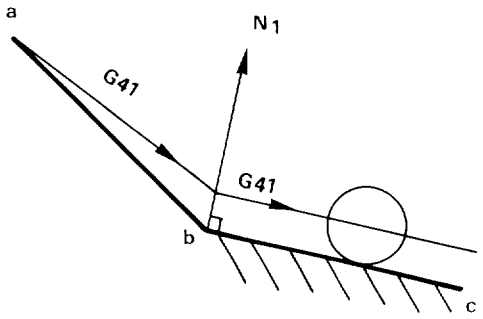
The system first calculates the offset normal to the start of the profile, and then calculates intersections parallel to the programmed profile, (straight lines and circles). It applies the offset to the left, (G41) or right, (G42) in the direction of travel. When the radius compensation is cancelled, by G40, (to straight turn or to face for example), then the cutting point reverts to point P.



5.7.1 - Entry - exit

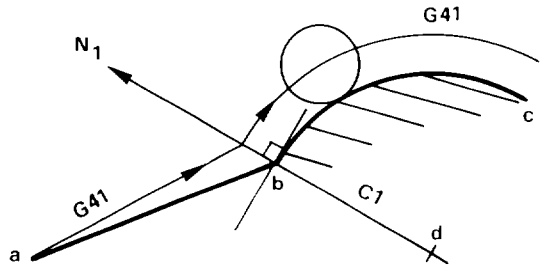
On entry into radius compensation mode, (G41 or G42), the programmed end point is modified so as to present the tool normal to the start point of the following move. The entry move MUST be linear.

- Entry into a straight line



N10 G1 Xa Za F
N20 G41 Xb Zb D10
N30 Xc Zc

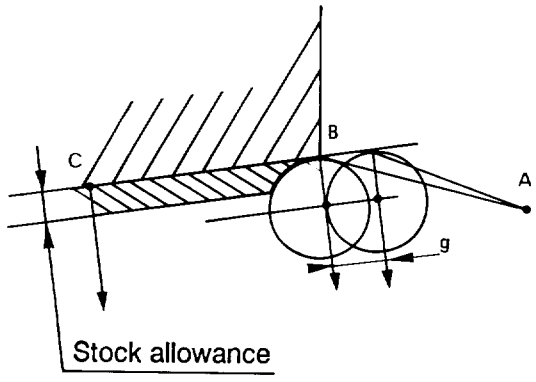
- Entry into a circle



N10 G1 Xa Za F
N20 G41 Xb Zb D20
N30 G2 Xc Zc Id Kd

The path ab must not be a circle.

- Precautions during entry



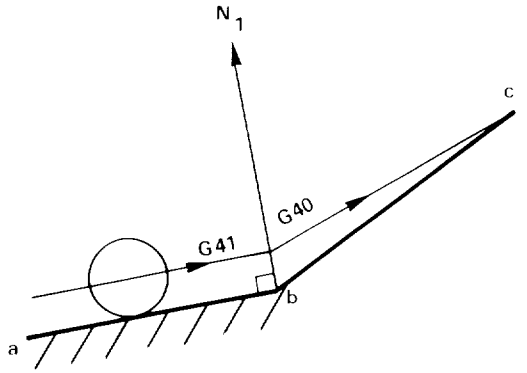
For entries onto paths which will have a stock allowance, the programming of point B can involve the unintentional removal of material, as shown in the figure.

Allow for a clearance (g) equal to or greater than the tool radius, back along the intended trajectory.

– Exit

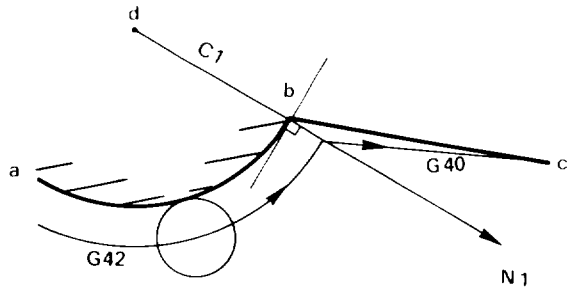
On exit from compensation mode, the last compensated move will finish at the normal to the programmed point. Then the radius compensation will be removed during the last move, (upto the G40 code). Once again, like the entry move, the exit move MUST be linear.

– Exit from a straight line




```
N10 X Z F
N20 D15
...
N100 G41 Xa Za
N110 Xb Zb
N120 G40 Xc Zc
```

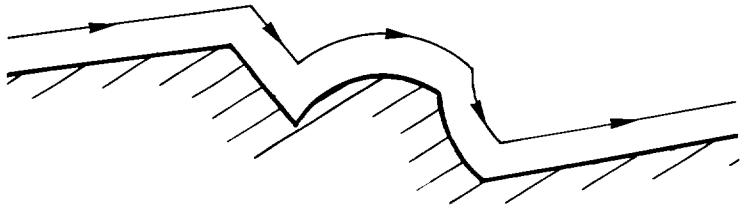
– Exit from a circle



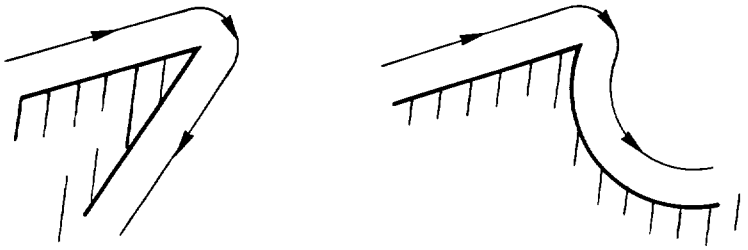

```
N10 F
N20 D16
...
N100 G42 Xa Za
N110 G3 Xb Zb Id Kd
N120 G40 G1 Xc Zc
```

5.7.2 – Consecutive paths

- All geometric elements (straight lines or circles) can be connected in radius compensation mode.



- When the compensation is "external" and the angle formed by two geometric elements is large, (greater than 120° for line–line sequences and greater than 90° for line–circle, circle–line and circle–circle sequences), then the system generates a connecting circle, or "roll round". If in block by block mode, the roll round will be appended to the current move.



5.7.3 – Points of note, with tool and radius compensations

- The presence of D0 – M0 – M1 in radius compensation mode.

D0 must not appear inside a radius compensated sequence of blocks.
Similarly, M0 or M1 must not appear inside a compensated sequence.

- G52 is not allowed in radius compensation mode.

- Changing a compensation number, (D), or wear compensation value :

  . changing a compensator number :
  New compensation values are taken into account as and when the axes are next programmed.
  New radius compensation values will be taken into account only after a G40 and a new call, (G41 or G42).

  . changing a compensation value :
  Compensation values can be changed at any time. New values will be taken into account as and when their respective axes are next programmed.

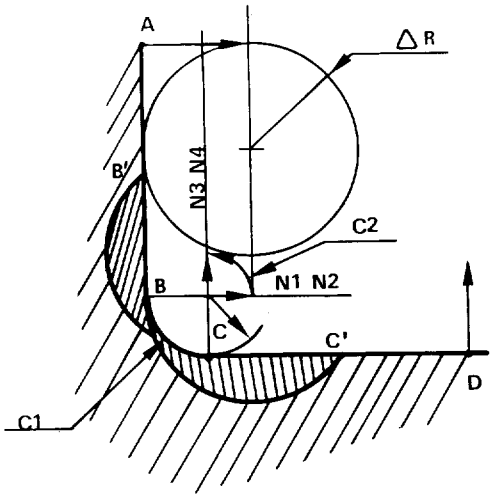
- Due to "look ahead" reasons, it is not possible to work with radius compensation in MDI mode.

5.7.4 – Practical compensation limitations

The tool radius should be equal to or less than the smallest radius programmed in G2 or G3. The examples which follow describe two common errors of programming in which the relationship between workpiece profile and real tool radius has not been fully appreciated. The system will detect this and display error message 149 if the error is not inhibited in machine parameter P7, word 1, bit 4.

. ΔR greater than the programmed circle

Use of a tool nose radius value greater than the radius of the smallest programmed circle. The programmed path is A,B,C,D.



The system first calculates a path parallel to AB, and at point B, constructs the normal (N1–N2) to create a tangential entry condition as required by the programmed circle, BC.

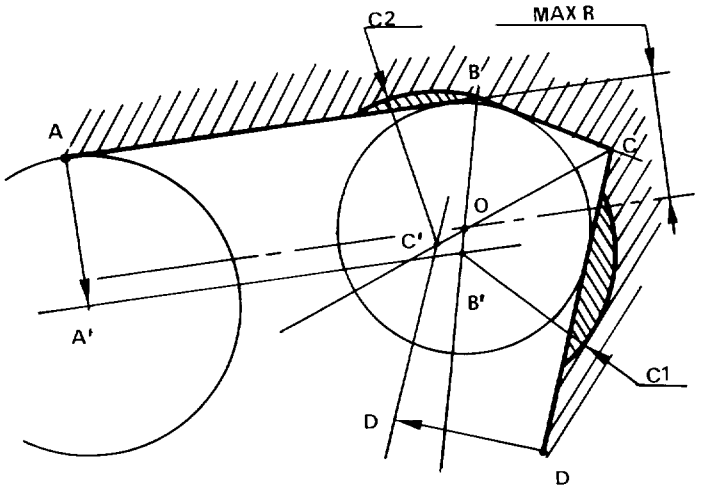
Similarly, the normal N3–N4 is constructed to the following linear block, CD, to give a tangential exit condition.

As true mathematical conditions for tangency have been created, so the system will generate the connecting circle, C2, in the counter clockwise direction, as programmed.

The figure clearly shows that this was unintentional, and how care should be taken to avoid having ΔR > radius C1.

. ΔR too big for small linear moves

Use of a ΔR value which is too large in terms of the small moves to be carried out. Again, the programmed path is A,B,C,D.



In considering the programmed path ABC, the system will calculate the required move as A'B'. The centre of the tool will be positioned to B', which is located on bisector $\widehat{ABC}$.

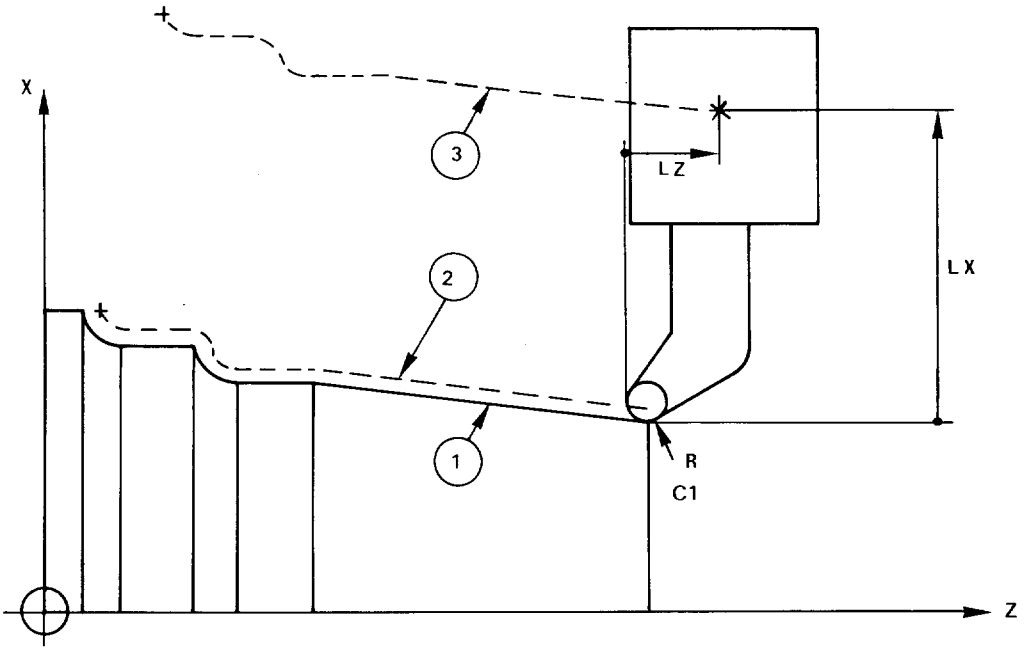
A similar consideration of the path BCD, will produce the move B'C'.The tool will be positioned to C', located on bisector $\widehat{BCD}$.

The successive positions will thus be A'B'C'D'.

The figure clearly shows that when the tool is first moved to B', it will already have unintentionally undercut the job. The following position, C', compounds the error, before exiting along the correct profile.

The shaded areas at C1 and C2 are wrong. In fact, the maximum ΔR tool nose radius compensation permitted can be calculated from the value of the projection of bisector BO on AA'. (O is the point of convergence of bisectors BB' and CC').

- EXAMPLE : Tool and nose radius compensation association.

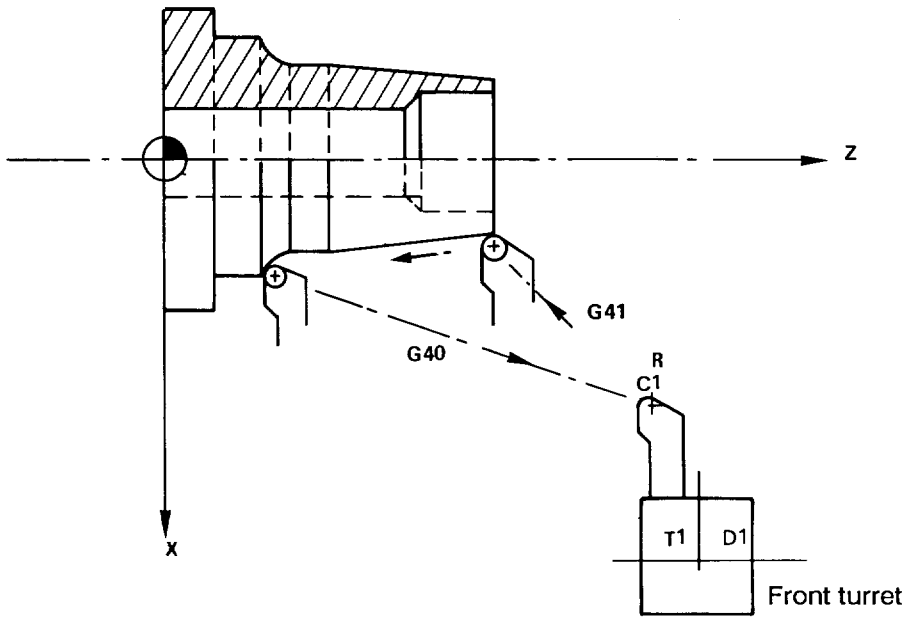


1 Programmed path.
2 Tool centre path.
3 Turret centre path.

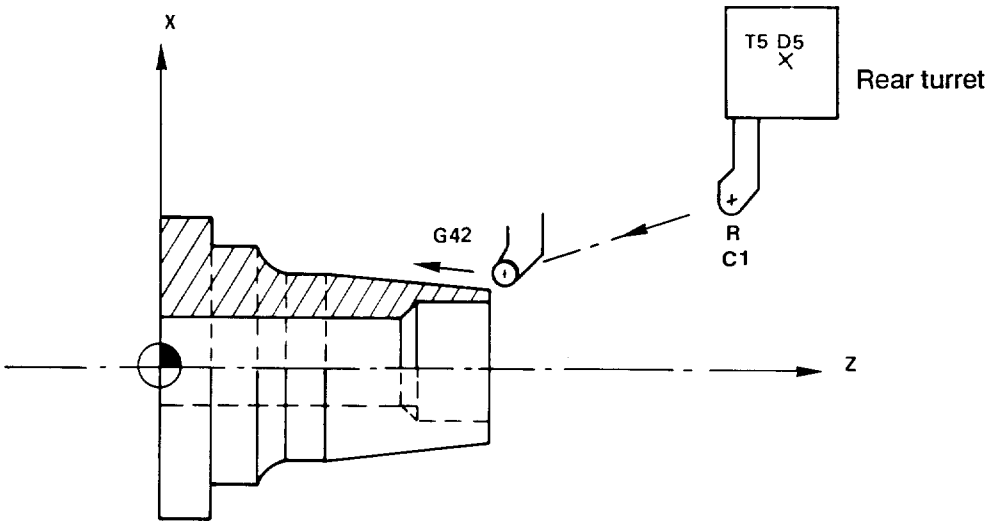
5.7.5 - Radius compensation with one or two turrets

- Lathe with a single turret

. Front turret

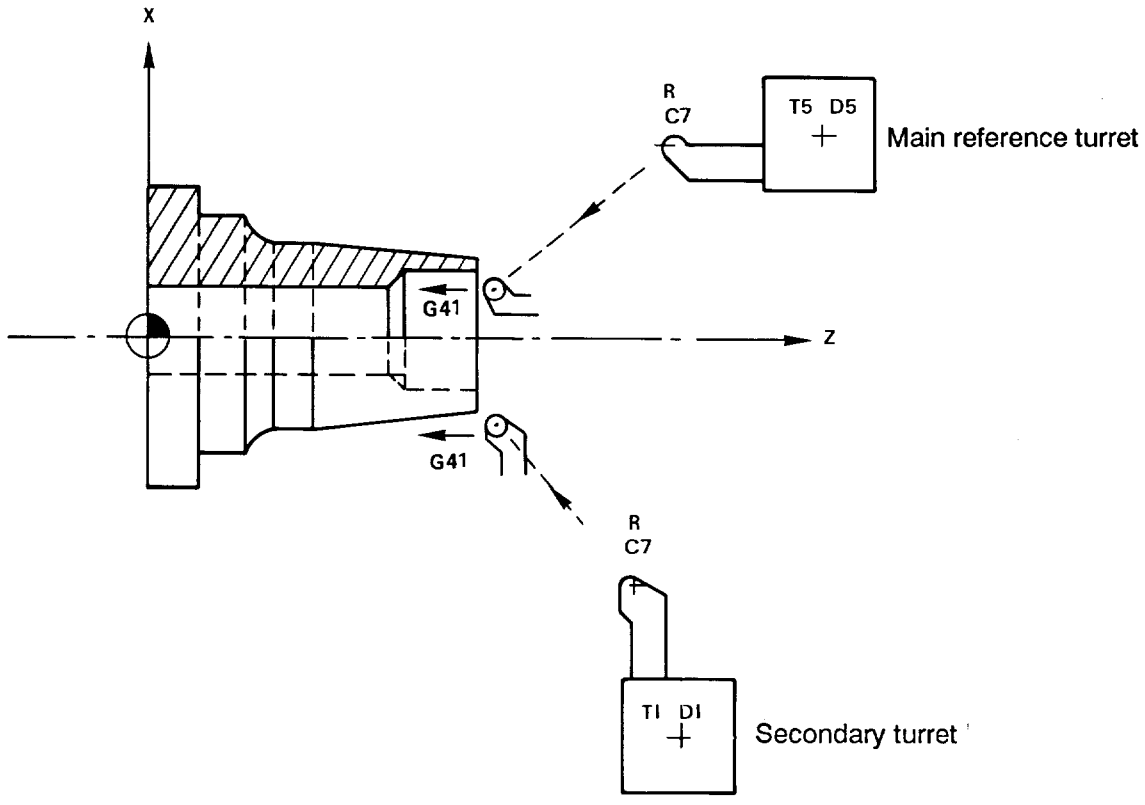


. Rear turret

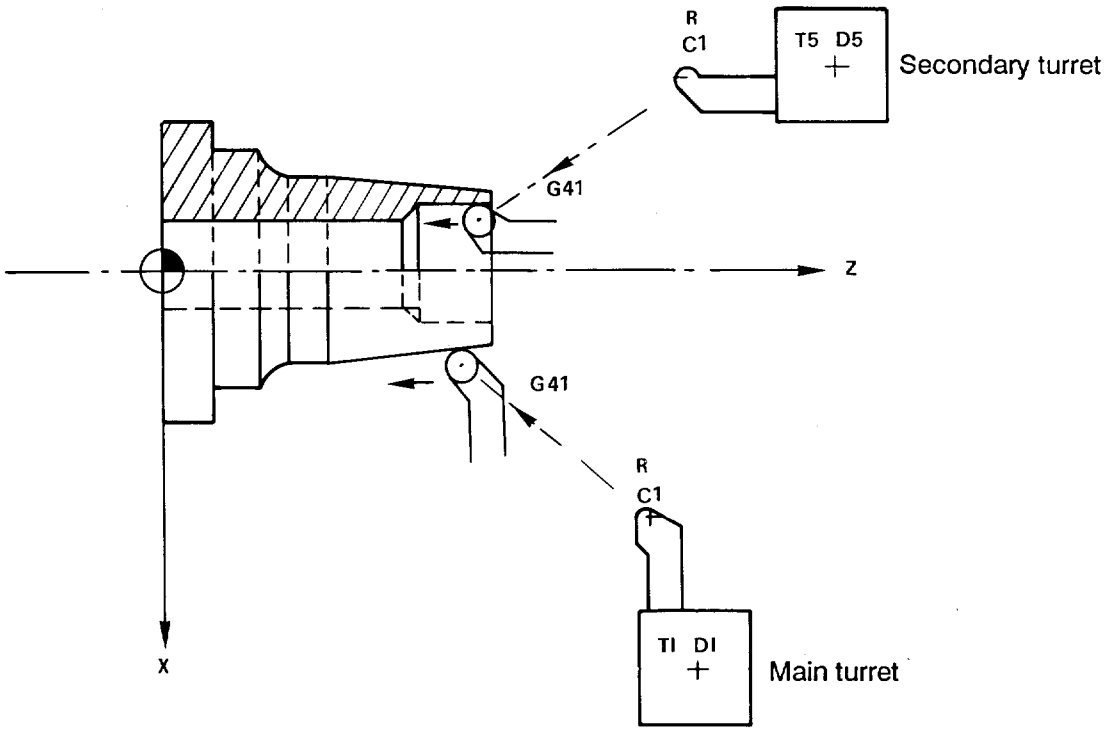


– Lathe with twin turrets

. Main turret at the rear



Main turret at the front



NOTES

6.1 – ROUGH TURNING/FACING CYCLE : G64

This cycle is used to rough turn a workpiece in paraxial mode, (ie. in the X or Z direction,) from the definitions of the blank and finish profiles.

6.1.1 – Programming syntax

G64	Nn	Nm	I±..	K±..	{P..} {R..}	cycle call block
	Xa	Za				
	Xb	Zb				
	Xc	Zc				definition of the blank profile
	Xd	Zd				
G80	Xe	Ze				end of the blank definition

6.1.1.1 – Cycle call block

The G64 function is followed by Nn and Nm. These two sequence numbers give the location in the part program of the finished profile definition. The order in which Nn and Nm are programmed in the G64 call block, will dictate the progression in which the area clearance work is to be carried out. It can either be the order in which these blocks are programmed, (example 2), or the reverse, (example 1).

Blocks Nn and Nm must both contain, both X and Z dimensions. (This allows starting from either end.)

The finish profile definition must be less than 50 blocks long.

{P..} the depth of each cut in the X direction.
{R..} the depth of each cut in the Z direction.

I.. the stock allowance in the X direction
K.. the stock allowance in the Z direction.

The stock allowances are optional, but if present, they are assigned to the finished profile. They must then be given values, the sign of which will dictate the direction of the stock allowance.

6.1.1.2 – Definition of the blank profile

The X and Z dimensions included between the start code, (G64), and the end code, (G80), define the blank profile. Machining progresses from the first point defined on the blank profile towards the last.

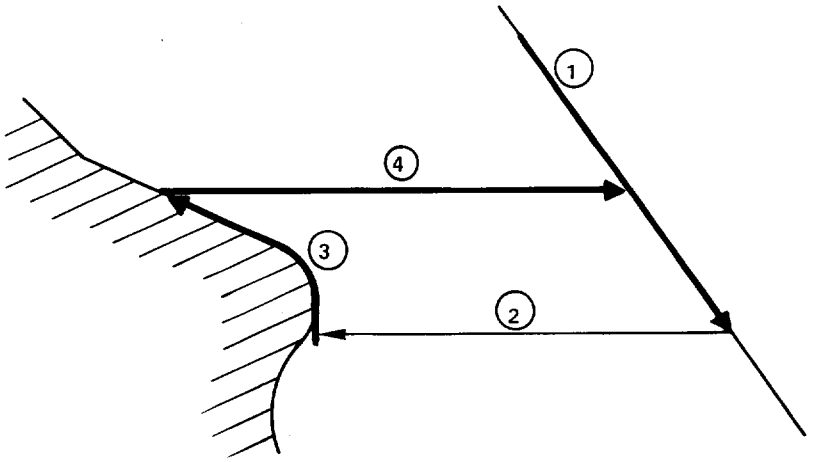
Unlike the finish profile definition, only straight lines are allowed, in the blank profile definition. (ie. G2 or G3 or GPP programming is not allowed.)

Auxiliary functions can be programmed and the depth of cut can be modified in the blocks which define the blank profile $\left\{\begin{matrix} P.. \\ R.. \end{matrix}\right\}$

6.1.2 – Description of the roughing cycle

Machining is carried out between the blank and finish profiles, (bearing in mind the possible stock allowance). Re–entrant parts of the profiles are left unmachined. At the end of the cycle the system is put into G0 status.

- Description of a cut

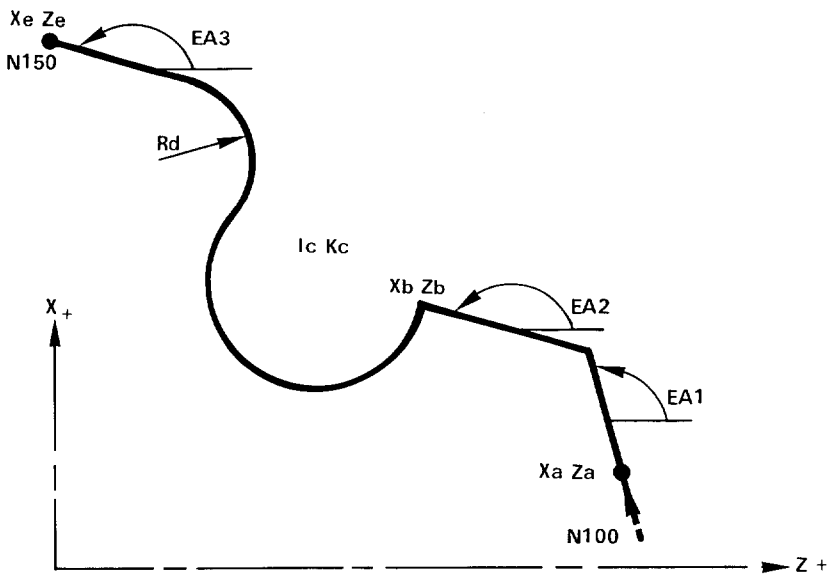


1 Rapid movement.
2 Rough turning at feedrate.
3 Continue along the finish profile at feedrate.
4 Rapid return.

6.1.3 – EXAMPLES :

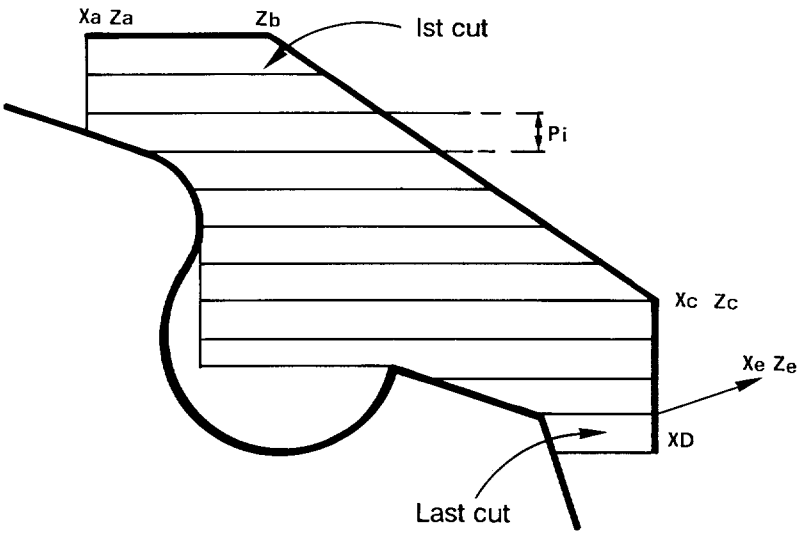
(Finish profile description)

N100	G1	Xa	Za	
N110	ES	EA1		
	EA2	Xb	Zb	
N130	G2	Ic	Kc	
	G3	Rd		
N150	G1	EA3	Xe	Ze



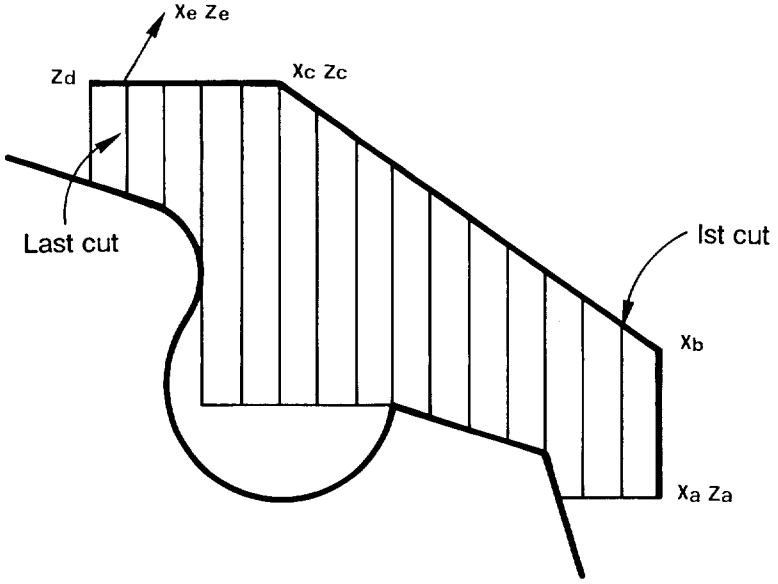
Example 1 :

N200	G64	N150	N100	Pi
N210	G1	Xa	Za	
N220	Zb			
N230	Xc	Zc		
N240	Xd			
N250	G80	Xe	Ze	



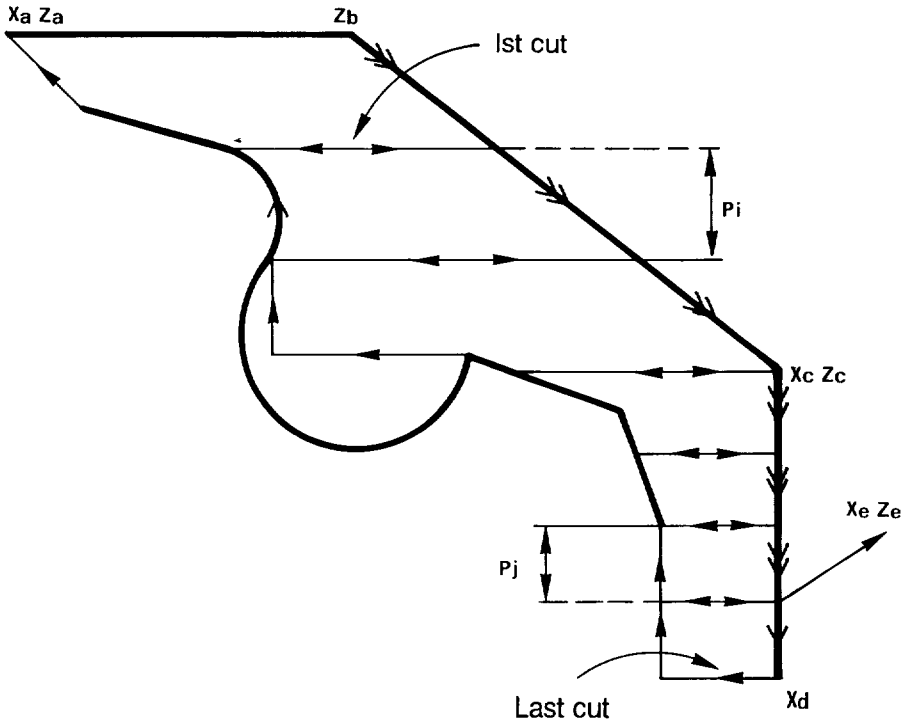
Example 2 :

```
N200  G64  N100  N150  Ri
N210  G1   Xa    Za
N220  Xb
N230  Xc   Zc
N240  Zd
N250  G80  Xe    Ze
```



Example 3 :

```
N200  G64  N150  N100  Pi
N210  G1   Xa    Za
N220  Zb
N230  Xc   Zc
N240  Xd   Pj
N250  G80  Xe    Ze
```



6.2 – GROOVING CYCLE : G65

6.2.1 – Programming syntax

G65 Nn Nm EA.. $\begin{Bmatrix} P.. \; Z\pm.. \\ R.. \; X\pm.. \end{Bmatrix}$ I±.. K±.. Q.. EF..

The G65 function is non–modal and is designed to machine the re–entrant profiles omitted by the G64 cycle.

Nn, Nm : these two sequence numbers specify the finish profile definition, which must include a portion on either side of the area to be machined.

As in the G64 cycle, the order in which Nn and Nm are programmed in the G65 call block, gives the direction in which the clearance work is to be carried out.

Blocks Nn and Nm must both include both X and Z dimensions. (This allows starting from either end.)

The finish profile must be less than 50 blocks long.

EA..	Penetration angle into the groove.
$\begin{Bmatrix} P.. \\ R.. \end{Bmatrix}$	the depth of each cut in the X direction. the depth of each cut in the Z direction.
$\begin{Bmatrix} X.. \\ Z.. \end{Bmatrix}$	Boundary of the zone to be cleared. (See cycle description). Either X or Z.
I.. K..	stock allowance in the X direction stock allowance in the Z direction.

The stock allowances are optional, but if present, they are assigned to the finished profile. They must then be given values, the sign of which will dictate the direction of the stock allowance.

Q — At the end of each cut, the positioning move to the beginning of the next cut can, if required, be performed in two stages :

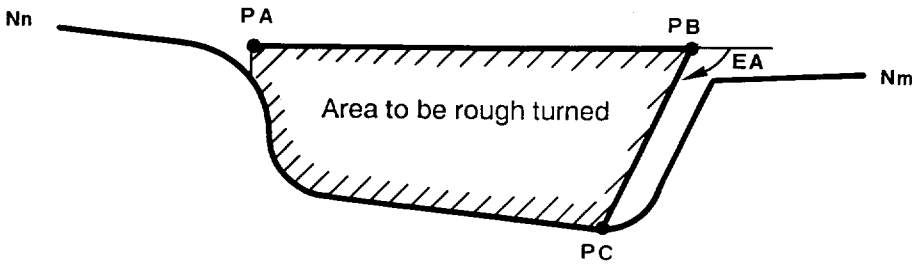
- the first stage at rapid, (G0), up to a distance Q from the beginning of the next cut.
- the second stage, at the previously defined feed rate, up to the beginning of the cut next.
  (If Q is ommitted, these two stages are merged and performed at feed rate.)

EF — The penetration rate into the material. If ommitted, it is taken as the currently active feed rate.

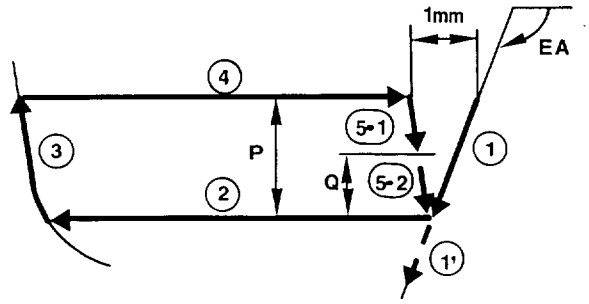
6.2.2 - Description of the grooving cycle

The area to be machined is bounded by the finish profile and by two straight lines linking the following three points :

- the X or Z boundary dimension programmed in the G65 cycle (PA),
- the start point of the cycle, PB. (ie. the last point programmed before the G65 function call),
- the intersection point of a straight line, at angle EA.., from PB, to the finished profile, (PC).

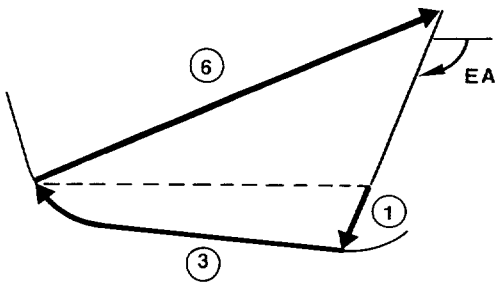


- Description of a cut

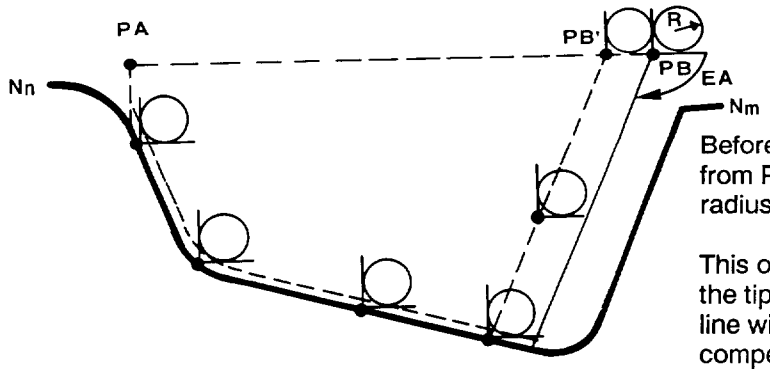


1 Feed in, along angle EA
2 Turn (or face).
3 When the profile is met, continue along it until clear.
4 Return at rapid to 1 mm from the starting point.
5–1 Rapid toward the start of the next cut, upto the Q limit. (Optional)
5–2 Move to the start of the next cut, at feed rate.
6 At the end of the last cut, rapid return to the start point of the cycle.

- Description of the last cut



- Offset to the penetration straight line to take account of the tool radius



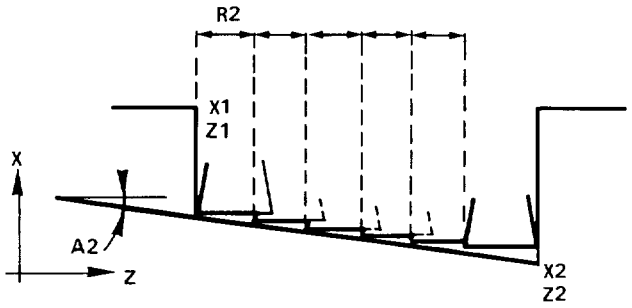
Before the start of the cycle, there is a rapid move from PB to PB', of a value equal to, the tool-tip radius * (1 + tan(180-EA)/2)

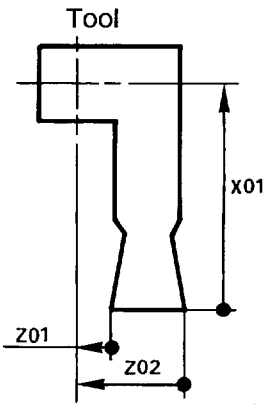
This offset from the penetration straight line allows the tip to be tangential to the penetration straight line without carrying out a complete tool nose radius compensation.

6.3 - PLUNGE GROOVING CYCLE : G66

This function enables a groove to be machined by successive plunge cuts.

Plunging a longitudinal groove





Compensators used : D01 = X01 Z01
D02 = X01 Z02

Program :

N100	G0	D01	X1	Z1			
N110	G66	D02	X2	Z2	R2	EA2	EF

- The initial block, N100, rapids the tool to the start point of the cycle. If the bottom surface of the groove is tapered, then the start point should be at the "shallow" end of the groove. The appropriate tool compensation is activated.

- In the following block, the plunge grooving cycle, (G66), is declared, together with the coordinates of the opposite corner of the groove, plus the alternate tool compensation, plus a plunge pitch and any dwell, (EF2.2), required at the bottom of each plunge.

When the bottom of the groove is flat, the programming of its angle is optional.

Dwell programming is also optional.

NOTE :

Widths of cut are uniformly spread over the entire width of the groove. The system will automatically adjust the value of the programmed pitch to ensure that this is the case.

6.4 – WOODPECK DRILLING CYCLE : G83

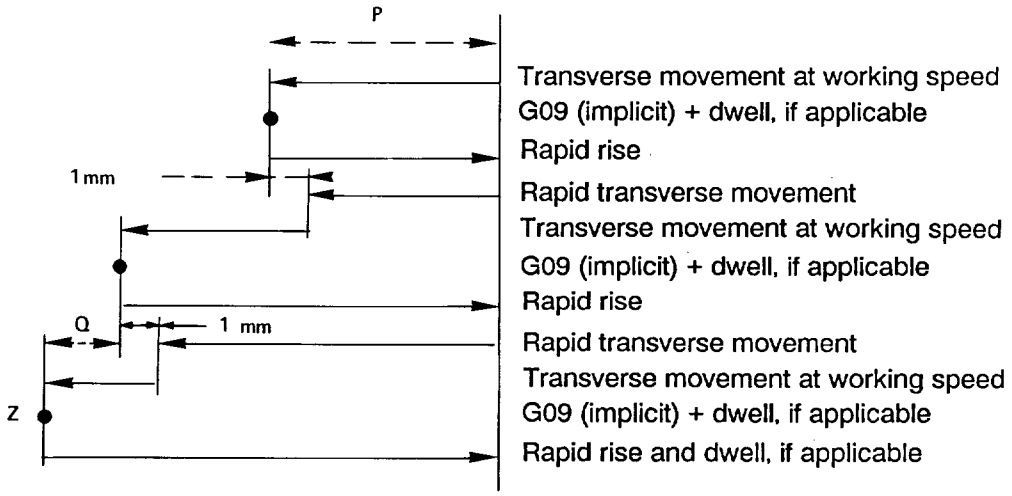
This modal cycle is used to make deep axial or radial holes with a turret mounted drill.

- Block format

  G83 X±.. Z±.. P.. Q.. F.. EF.. G4F..

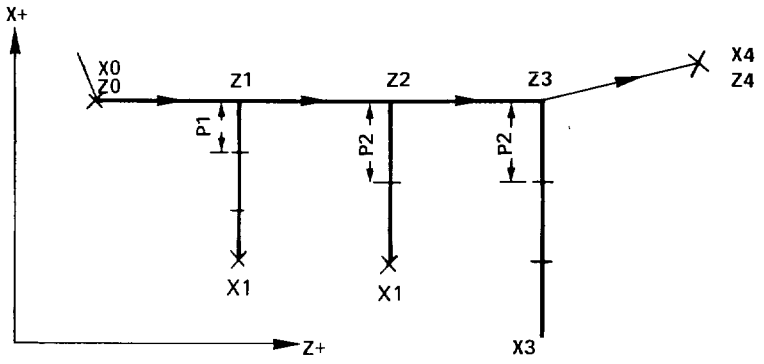
  X–Z.. Dimension at the bottom of the hole
  P.. Value of the first penetration
  Q.. Value of the last penetration (optional)
  F.. Feed rate in mm/min
  EF.. Dwell at the end of each penetration (optional)
  G4F.. Dwell after withdrawal of the last cut (optional)

- If Q is ommitted, the penetration pitch is constant.



EXAMPLE :

G0	X0	Z0		
G83	X1	Z1	P1	F
		Z2	P2	
	X3	Z3		
G80	X4	Z4		



6.5 – CHIP–BREAK DRILLING CYCLE : G87

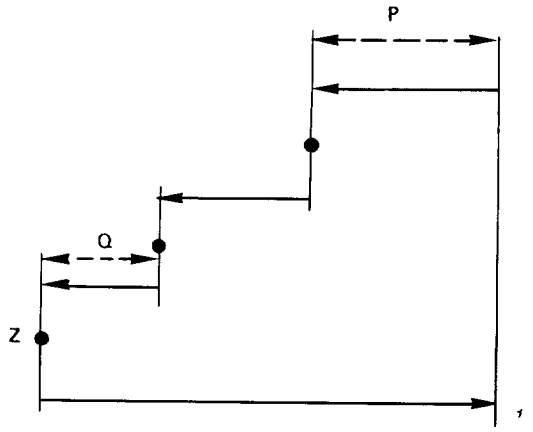
This modal cycle is used to make axial or radial drilled holes with a similar action to the G87 (woodpeck) cycle, but with a chipbreaking halt of feed, instead of a withdrawl, after each peck.

Block format

G87 X±.. Z±.. P.. Q.. F.. EF.. G4F

X..	Z..	Dimension at the bottom of the hole
P..		Value of the first penetration
Q..		Value of the last penetration (optional)
F..		Feed rate in mm/min
EF..		Dwell at the end of each penetration (optional)
G4F..		Dwell after final withdrawal of the tool (optional)

– If Q is ommitted, the penetration pitch is constant.



First feed move

G09 (implicit) + dwell, if applicable

Next feed move

G09 (implicit) + dwell, if applicable

Next feed move

G09 (implicit) + dwell, if applicable

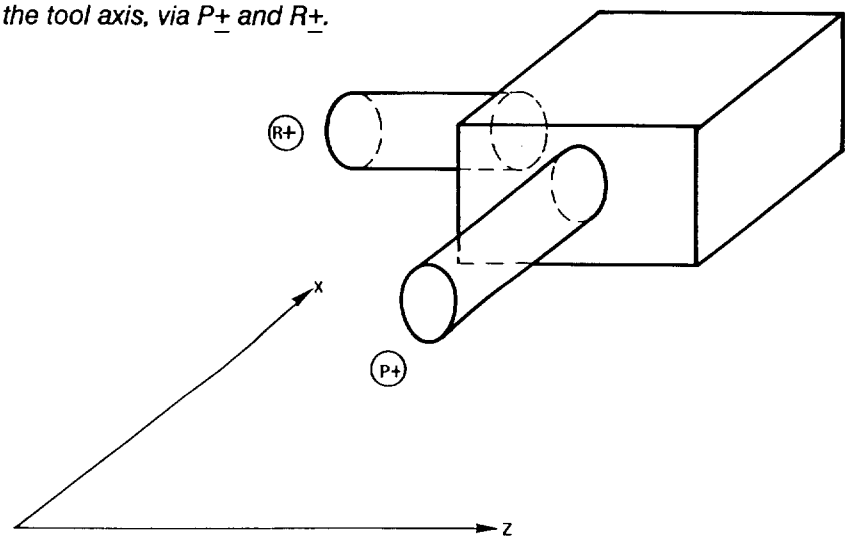
Rapid retract + dwell, if applicable

NOTE :

It may be possible for machines with interchangeable toolholders or driven tooling to execute the cycles G83 and G87 in either X or Z.
In either case, the tool axis must always be parallel to X or Z.

In these cases, G16 is used to define the tool axis, via P± and R±.

EG. G16 P+



When first switched on, the system is initialized to G16 R+

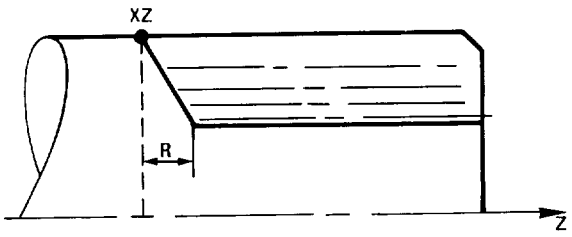
6.6 – THREADING CYCLE : G33

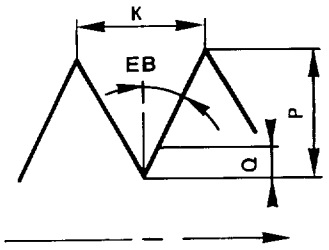
By programming a single G33 block, it is possible to define and execute a complete cylindrical, or tapered threading cycle, (or scroll), with constant pitch and decreasing depth of cut, (giving a constant metal removal rate).

The maximum thread pitch is 250 mm.

6.6.1 – Threading block format

G33 X± Z± K EA EB± R P Q F S





X, Z – the end of thread, coordinates. (**mandatory**) Can be absolute or incremental values

EA
- EA the cone angle. (**optional**) By default, EA is 0°; ie. a parallel thread.
- If EA is 90° : then a scroll thread is defined.
- −45° ≤ EA or EA ≤ 45° : Z major axis (threading axis), X minor axis (penetration axis).
- EA > 45° or EA < −45° : Z minor axis (penetration axis), X major axis (threading axis).

K – major axis pitch, (not the pitch along the taper). (**mandatory**) Unsigned value, 250 mm max.

P – total depth of thread, Q included. (**mandatory**) An unsigned value.

F – number of starts. (**optional**) Single start by default, (maximum 9).

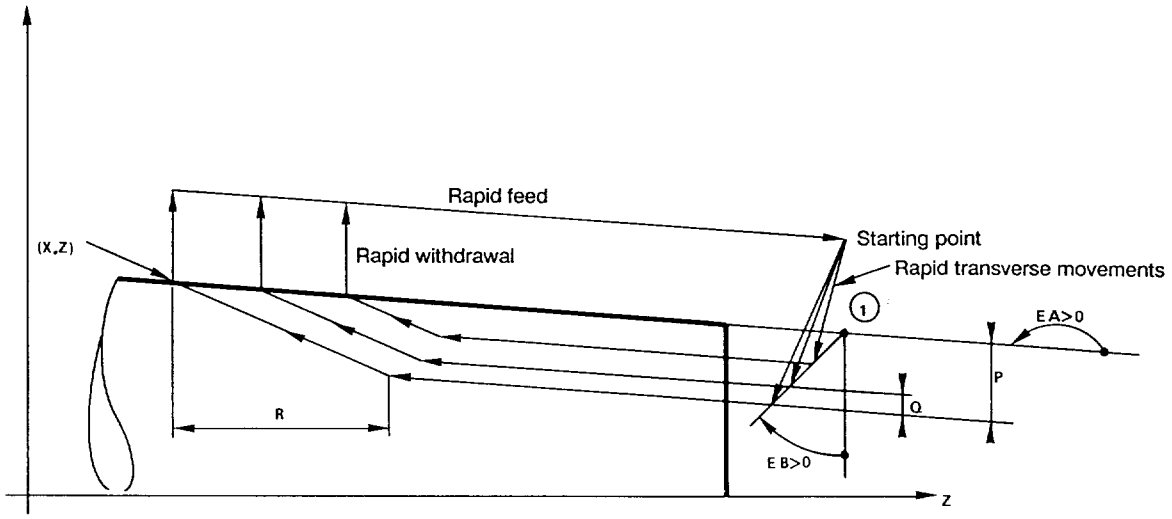
Q
- depth of the final cut (included in P). (**optional**)
- an unsigned value. With flank progression, Q is measured along the flank at angle EB.
- if Q is given but with a zero value, a sizing cut will be made.

EB
- flank progression angle. (**optional**) By default, EB is 0 : straight penetration
- the penetration flank is determined by the sign of EB :
  EB > 0 penetration is in the direction of threading
  EB < 0 penetration is in the opposite direction
- values are in thousandth of a degree (EB 3.3)

S – number of metal removal passes, ie. excluding a sizing cut. (**optional**) By default S = 1.

R – length of the pullout taper on the major axis. (**optional**) Unsigned value, 0 by default.

6.6.2 - Representation of dimensions



An error is reported if the start point, (ie. the last position prior to the G33 code,) lies within the workpiece profile, as defined by the end point and the cone angle. (point 1 on the figure).

6.6.3 - Breakdown of a multi-start threading cycle

If there are n starts :

1st cut on the 1st start

⋮ ⋮

1st cut on the nth start

2nd cut on the 1st start

⋮ ⋮

2nd cut on the nth start

etc.

6.6.4 – Programming threads.

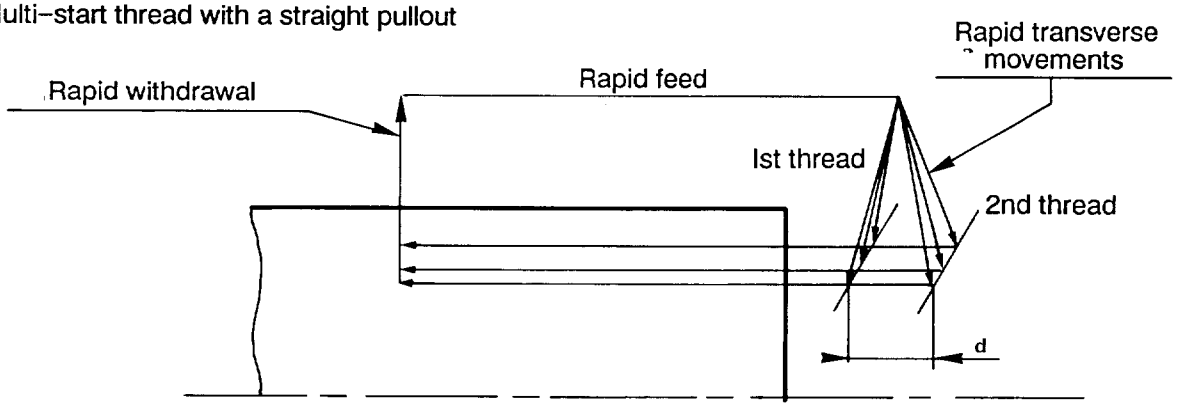
G33, Non–modal

Addresses	Mandatory data, in a G33 block	Values taken by default
X	yes	
Z	yes	
K	yes	
P	yes	
Q	no	no sizing cut
R	no	R = 0 straight pullout
EA	no	EA = 0 parallel thread
EB	no	EB = 0 straight penetration
F	no	F = 1 single start
S	no	S = 1 single cut to depth

* Modal functions present before block G33 are restored at the end of the threading cycle for the following blocks. There are no modal parameters set by G33.

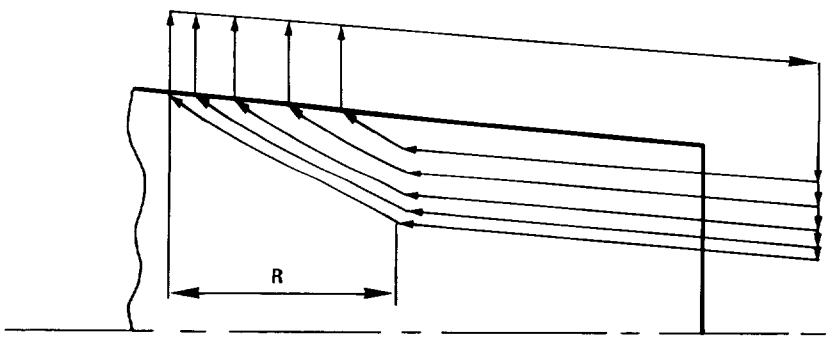
6.6.5 – Miscellaneous examples

– Multi–start thread with a straight pullout

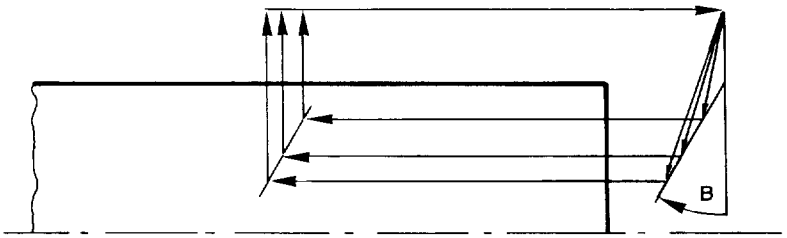


The offset required for the second start is always applied in the opposite direction to the direction of threading. (This increases the acceleration distance.)

– taper threading, a single start thread with tapered pullout and straight penetration.



- parallel threading, a single start thread with flank penetration and straight pullout.



NOTE :

This is not good practice. There should be a tapered pullout.

6.7 - CONTINUOUS THREAD CUTTING : G38

When G38 is used, the axes are automatically controlled by the position of the spindle for a number of consecutive linear blocks.

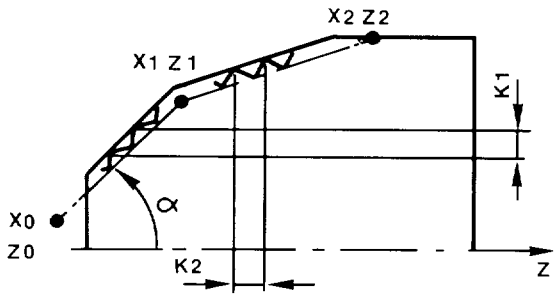
G38 X Z K

X and Z, absolute or incremental position of the thread end point.

K, the thread pitch must be programmed in the first block. It can be modified in following blocks.

The thread pitch is assigned to the major axis of the block, (ie. the axis with the longest travel).

In order to retain feed synchronization with the spindle, auxiliary functions must not be programmed within the set of blocks defining the threaded profile. Do not use constant cutting speed. (The spindle speed changes, with each new depth of cut in the X direction, would cause a loss of pitch synchronization.)



G	X_0	Z_0	
G38	X_1	Z_1	K_1
	X_2	Z_2	K_2

Cycles can be built up with sub-program loops.

6.8 – RIGID TAPPING CYCLE : FUNCTION G84

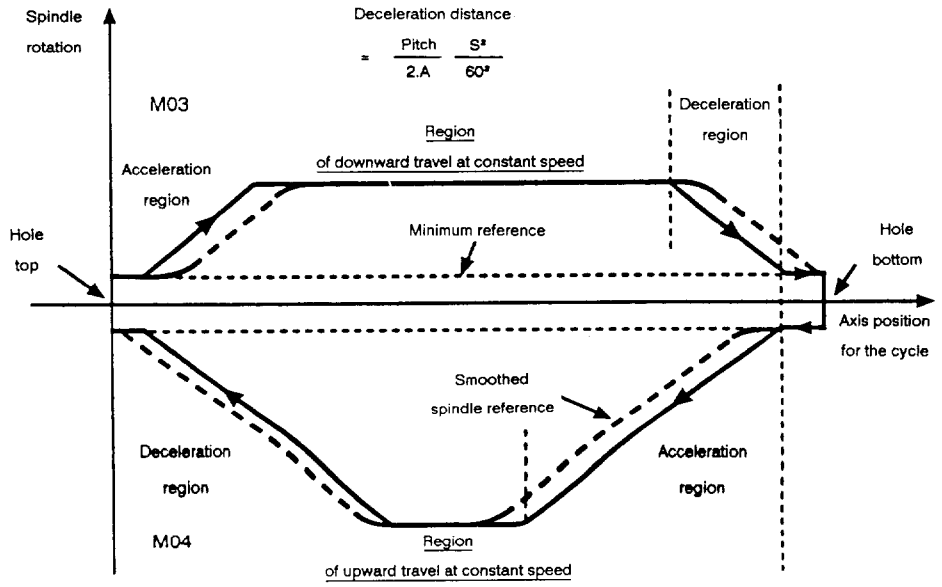
With the rigid tapping function, the tool axis feed rate is slaved to the spindle rotation.

IMPORTANT : If the system includes this option, do not use the normal G84 tapping cycle.

At the beginning of the entry phase, the tool axis is coupled to the spindle rotation : the spindle starts and stops rotating during tapping according to the deceleration distance. The tool axis following error is cancelled during this cycle.

This cycle includes eight phases :

- Indexing of the tool or chuck in the case of tapping on center in the Z axis : positioning must be programmed in a previous block.
- A phase during which the spindle speed and feed rate are increased, with the two coupled electronically.
- An entry phase at constant speed.
- A deceleration phase to reach the hole bottom. The spindle reference is proportional to the distance remaining to be travelled.
- A spindle rotation reversal phase at the bottom of the hole. In the hole bottom area, the spindle is kept rotating to prevent it from stopping completely.
- An acceleration phase.
- An exit phase at constant speed.
- A deceleration phase up to the hole top ; the spindle is not completely stopped on arrival.



REMARK :

For rigid tapping not on the part center line in the Z axis, the lathe must be equipped with a measured, indexable chuck and a measured, indexable tool holder spindle.

Programming

The rigid tapping cycle is called by modal function G84 and is cancelled by functions G64, G65, G66, G80, G83 and G87.

General format :

G84 X+053 or Z+053 K053 EK053

This block must be preceded by a positioning block.

X or Z : hole bottom dimension (X for tapping perpendicular to the Z axis, Z for tapping parallel to the Z axis).

K : tapping pitch in mm (K is programmed to specify leadscrew tapping).

EK : spindle in/out speed ratio. If EK is not included in the block, a ratio of 1 is used as default.

Programming example : rigid tapping on center line in the Z axis.

N50 M3 M41 S 400 X Z5	Positioning
N60 G84 Z-20 K1 EK4	Tapping to Z = −20, pitch of 1, exit travel four times faster than entry travel.
N70 G80 G X 40 Z5	Retraction

CAUTION :

Action on FEED STOP prematurely terminates the cycle before reaching the bottom of the hole. Once actioned FEED STOP cannot be cancelled until the axis of the cycle has returned to the top of the hole, at which time the machine stops.

REMARK :

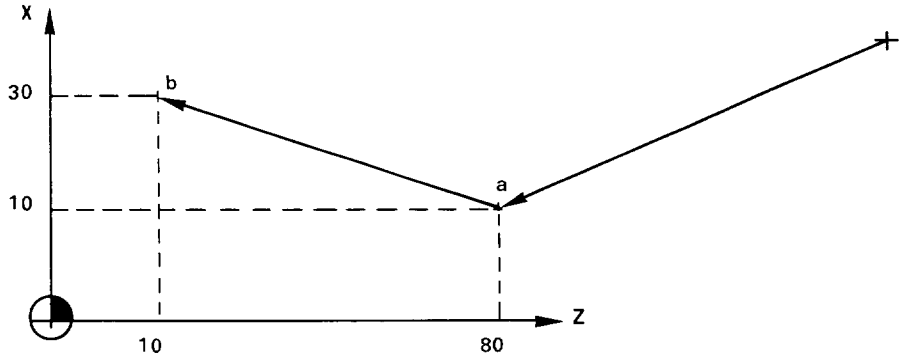
Rigid tapping can be carried out in several passes, but this requires programming several blocks, e.g. :

```
G X...  Z...  M3 M41 S400
G84 Z1 K... EK...
Z2
Z3
```

NOTES

7 - FEED RATE - DWELL - SPINDLE SPEED

7.1 - FEED PROGRAMMING



7.1.1 - Programming in mm/min

G94 Preparatory code which defines feedrates in mm/min.

F5.2 Feed value (from 0.01 mm/min to the max. traverse rate)

Both G94 and the value of F are modal.

```
G0              X10     Z80
G1      G94     X30     Z10     F500
```

Path ab is executed at 500 mm/min.

7.1.2 - Programming in mm/rev

G95 Preparatory code which defines feedrates in mm/rev.

F2.3 Feed value (from 0.001 mm/rev to 16mm/rev)

Both G95 and the value of F are modal. (G95 must be programmed before the F value).

- If F is programmed with a value greater than 16, the system will automatically reduce it, by continually halving the value until it lies between 8 and 16 mm/rev.
  eg, if F84 was programed, the system would reduce it thus: 84/2 --> 42/2 --> 21/2 --> 10.5 mm/rev.
  No error will be reported.

- With G95, the spindle must be rotating to obtain any axis feed motion.

EXAMPLE :

```
G0              X10   Z80   S120   M3
G1    G95       X30   Z10   F0.15
```

The path will be executed at 0.15 mm/rev.

NOTE :

- *Programming in mm/min and in mm/rev is associated with the preparatory functions G1 – G2 – G3.*
- *Function G0 is modal and suspends the action of F. All movements are carried out at rapid.*
- *In radius compensation mode (G41 – G42), the programmed feed rate (F), applies to the tip centre path. (see Fig. 1).*

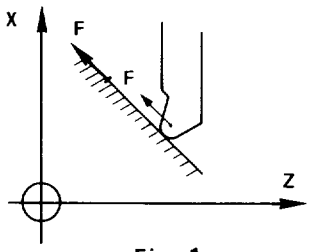


Fig. 1

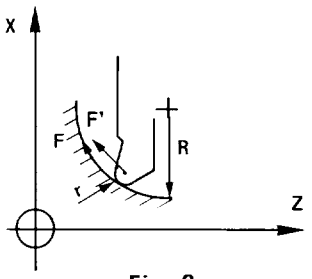


Fig. 2

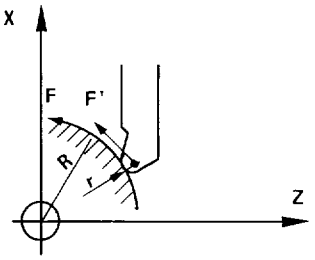


Fig. 3

When in circular interpolation mode, the feedrate at the point of contact may be higher or lower depending on whether the path is concave or convex.
To maintain a consistent feedrate over the workpiece profile, a feedrate F' must be programmed with the value :

$$F' : \quad \frac{F \text{ required at point of contact } x\ (R \pm r)}{R}$$

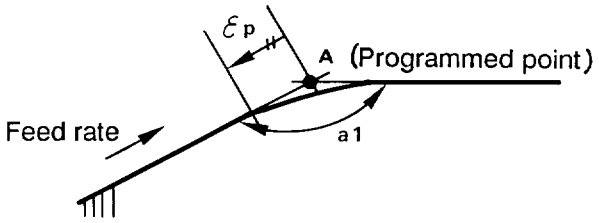
Figure 2 : Concave profile $F' = \dfrac{F \times (R - r)}{R}$

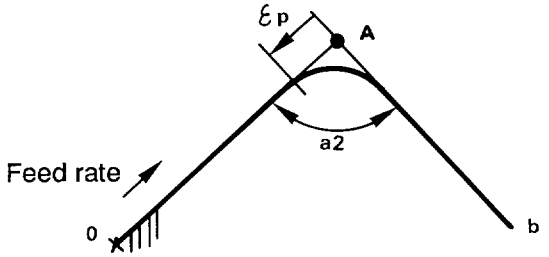
Figure 3 : Convex profile $F' = \dfrac{F \times (R + r)}{R}$

This modification is justified only if the result of operation $1 \pm \dfrac{r}{R}$ *differs significantly from 1.*

7.1.3 – Deceleration function G9

The axes servo following error, ∈p, is directly proportional to the programmed feedrate. Its "SMOOTHING" effect (at a constant feedrate,) increases as the angle α becomes more acute.





Bold line : The real path without G9, skids past "a"

Fine line : The real path with G9, actually goes through "a".

- G9 is used to absorb the following error ∈p at the end of a given movement before continuing to the next block. In doing this the axes drop down to zero speed.
- The programmed path is followed exactly.

The G9 function allows the next block movements to continue, provided that the current block dimensions have reached an "in position" window, (as pre–defined by the machine builder in parameter P22).

N1	G1	X0	Z0	
N2	G9	Xa	Za	LH figure
N3		Xb	Zb	

The slide will be decelerated along "0A" at a distance ∈p from "a" and will go through point "a".

N1	G1	X0	Z0	
N2		Xa	Za	RH figure
N3		Xb	Zb	

The slide will not be decelerated and will not pass through point "a". The curve described will be proportional to the feed rate and the included angle.

7.1.4 – Limit rates

The limit feed rates depend on the machine and are set up when the system is switched on. In linear mode, the minimum time for execution of a block is 0.08 seconds, irrespective of the rate programmed. In circular interpolation mode, the maximum tangential feed rate is proportional to the radius of the circle described by the centre of the cutter. It is expressed by the formula : Feedrate mm/min = 300*R in mm, giving a maximum angular speed of 1/20 radius/10 ms.

For example : a 3mm radius circle will be described at a MAXIMUM feedrate of 900 mm/min, irrespective of the programmed feedrate.

7.1.5 – Overspeed – function G12

If the machine–tool is fitted with a handwheel, then a G12 block will allow handwheel rotation by the operator to be superimposed on the normal program demanded feedrate. This can be of use to reduce "air time" during the approach to an un–machined surface.

Movement acceleration applied in this manner, is effective over the true programmed path (linear or circular).

The handwheel feedrate enhancements are enabled by programming G12 in the block concerned. This function is non–modal. The system continues with the following block when the demanded position is reached.

The value of the handwheel increment is defined, by the machine builder, in the 3rd and 4th words of machine parameter P13.

7.1.6 – Programming of Tangential Feed : Function G92R

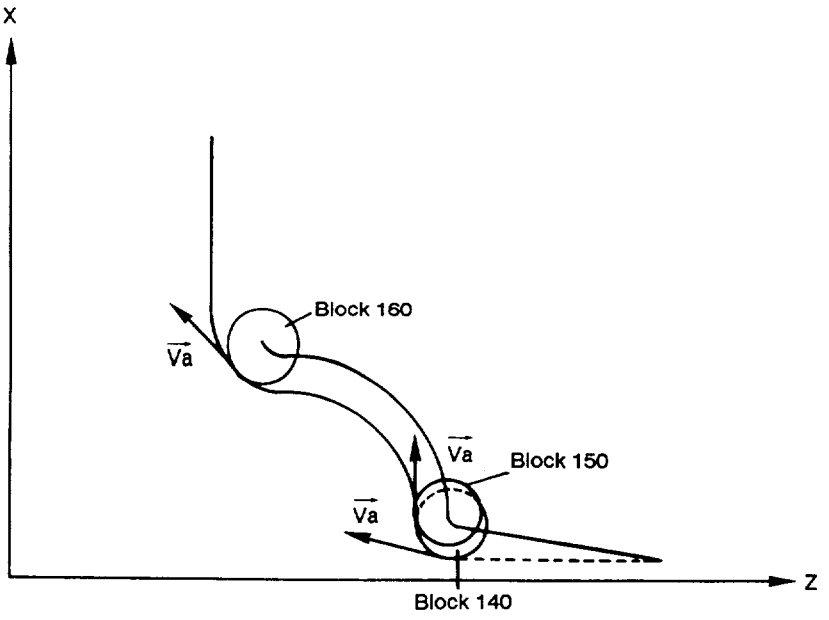
This function is used to apply the feed rate F.. to the programmed curve instead of to the tool center path.

It is used for tool radius compensation (G41, G42) and for programmed curves whose radius is higher than the value of the tool radius defined by R.

Function G92 R0 cancels programmed tangential feed.

EXAMPLE :

```
%
N100 T1 D1 M6                    Compensation D1: R = 10
N110 S800 M40 M3
N120 G0 G42 X Z100 F200
N130 G92 R10                     (Tangential feed rate programmed for curves whose radius exceeds
                                 tool radius R10)

N140 G2 X10 Z90 R10
N150 G3 X40 Z60 I10 K60          (Curves whose radius R is higher than the value of tool radius R
N160 G2 X60 Z40 R20              specified in block N130 : tangential feed is enabled)
N170 G1 X100
N180 G92 R0                      (Programmed tangential feed cancelled).
.
.
.
.
M2
```



NOTES :

- *When a connection circle is created automatically, the tool center feed on the circle remains the same as that of the previous block.*
- *Programming block G92 R... must not include any movements.*
- *Tangential feed is cancelled by a reset.*

7.1.7 – Programmed Acceleration Override : Function EG

This function is used to override the maximum acceleration tolerated on programmed movements by 1 to 100 percent. The maximum acceleration depends on the tool path and the accelerations assigned to the axes by machine parameters.

This function is modal and is forced to 100 by M02 or a reset. It is changed by programming a new value.

At power on, it is forced to 100.

- Programming : by function EG followed by an absolute integer between 1 and 100.

  The default value is 100.

  . If EG0 is programmed, it is converted to EG1 by the system,
  . If EG200 is programmed, it is converted to EG100 by the system.

  EG1 to EG99 are displayed on the current block page, but not EG100.

- Interpolation : in programmed movements, the acceleration override is always taken into account by the interpolations except in case of FEED STOP or trip of the feed stop security.

  In JOG, MOS or INTERV mode, acceleration override is inoperative.

7.1.8 – Special Feed Rate on a Fillet or Chamfer

A feed rate different from the machining feed rate can be programmed for cutting a fillet or chamfer programmed by EB+ or EB–.

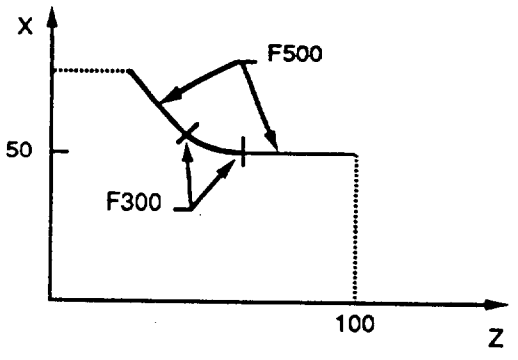
The feed rate is defined by address EF which is not modal.

It must be programmed after function EB. Otherwise, error 86 is displayed.

EXAMPLE :

```
N1  X 50  Z100  F500
N2  G1    Z50   EB10  EF300
N3  Z20  X 75
```

The fillet is cut at a feed rate of 300.



7.2 - PROGRAM DWELLS

G4 Dwell function

F2.2 Value of dwell, from 1/100th sec. to 99.99 secs.

The G4 function and the dwell value are non-modal.

Dwells programmed in blocks containing axis movements are active at the end of the block. The presence of F2.2 does not cancel the previously programmed feedrate.

(If G4 is followed by two F values in the same block, only the last one programmed will be effective.)

EXAMPLE :

N10		F500	500 mm/min feed
N20	G4	F1.5	1.5 sec dwell
N30			500 mm/min feed
N40	G95	F0.1 S600	0,1 mm/min or 0.1 mm/rev feed
N50	G0		Rapid feed

7.3 – PROGRAMMING THE SPINDLE

7.3.1 – Spindle programming in rpm : G97

The maximum programmable spindle speed is 32767 revs/min. (G97 S5)

The value of S is modal.

Upto three speed ranges may be defined, M40 to M42. (If an automatic range selection function is available, the range will be selected simply by programming S.)

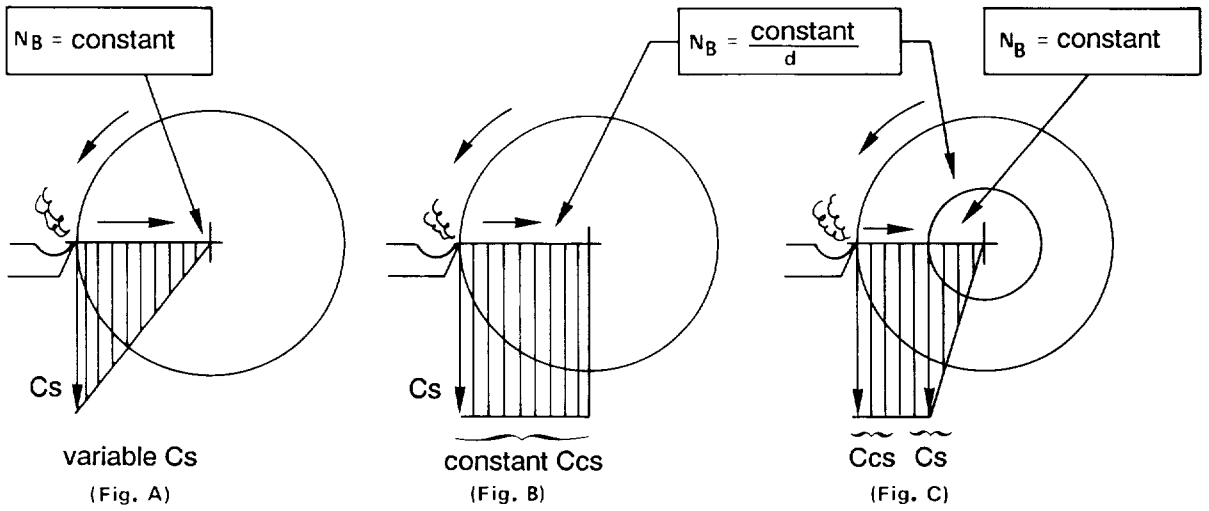
The G97 function is the default speed programming code.

NOTE :

G97 must always be accompanied by S, otherwise the system will generate an error message.

7.3.2 – Constant cutting speed (CCSPD) : G96

Reminder of what constant cutting speed (ccspd) does:



General equation for cutting speed

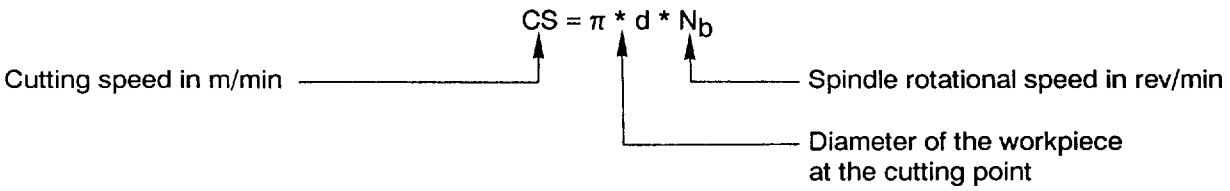


Figure A :

The spindle speed, (Nb,) is constant. Thus, the cutting speed decreases as the diameter decreases.

NOTE :

The cutting speed at the centre is therefore zero.

Figure B :

The cutting speed is kept constant (CCSPD). The spindle speed, (N_b,) will be inversely proportional to the diameter d.

NOTE :

When d = 0, the calculated spindle revs, N_b *will be infinite. This is obviously impossible to achieve, since the spindle can only rotate upto its maximum speed, eg 3000 rev/min.*

Figure C :

Same as B, but shows the effect of CCSPD in practice, whereby the spindle increases to its maximum revs, beyond which we revert back to the constant spindle speed and variable cutting speed of Figure A.

Format :

G96	X ± 5.3	S5
↓	↓	↓
CCSPD cancelled by G97	initial attachment diameter for calculating CCSPD	Cutting speed in m/min

Both the G96 function and the S value, are modal.

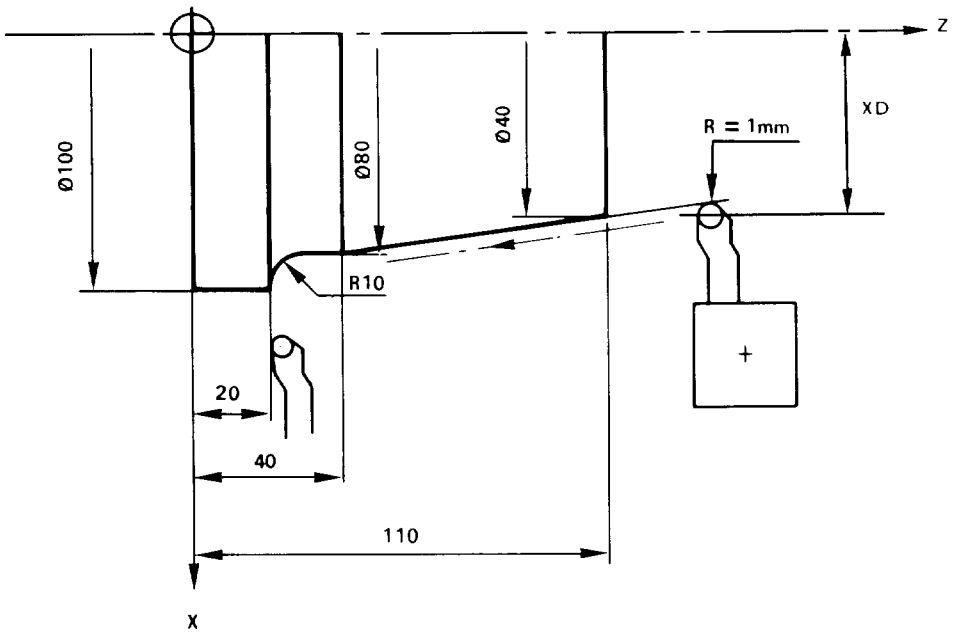
When CCSPD is called, the X axis must be programmed in the block with G96 or must have been in one of the blocks located between the last G52 function and function G96. (If X is left undefined, error 28 will be reported.)

Constant cutting speed is cancelled by a G97, with S5.

Obviously, CCSPD can only work with an analog spindle (S5).

When working with constant cutting speed, it is advisable to program the feed in mm/rev (G95), in order to obtain a constant swarf thickness. This will automatically take the varying spindle speed into account.

Programming examples



Example 1 : Workpiece profile programming

```
N20 S800 M41 M3
...
N80 T1 D1 M6
N90     G41          X37.144    Z115                    (X,Z machining starts)
N100    G96          X39.144                            S250
N110    G95    G1    X80        Z40                     F.150
N120                              Z30
N130    G2           X100       Z20        I100   K30
N140    G1                      Z0
N150    G97                                             S1000
N160    G40    G0    X150       Z200                    (withdrawal point)
```

N90 : positions the tool the to start of the constant cutting speed sequence.

N100 : Initializes the CCSPD calculation and starts the spindle at 2033 rpm., ie. 250000mm/(π*39.144). The cutting speed is calculated from the centre of the tool tip. To ensure that the cutting speed is right for the tip cutting edge, this corrected position should be programmed.

N150 : Cancels constant cutting speed, (G97 S1000), and fixes the spindle speed at 1000 rpm.

Example 2 : With tool dimensions included in the program.

Tool dimensions: X120 Z20

```
N20 S800 M41 M6
N80 G41 T1D1 M6
N90                    X157.144    Z135                (X,Z machining start)
N100      G96                                S250
N110      G1    G95    X200        Z60          F.150
N120                               Z50
N130      G2           X220        Z40          I100   K30
N140      G1                       Z20
N150      G97                                S1000
N160      G     G40    X270        Z220                (withdrawal point)
```

N100 : Same note as for example 1.
The cutting speed is calculated from the centre of the tool tip.

N150 : Cancels constant cutting speed, and leaves the spindle rotating at 1000 rpm.

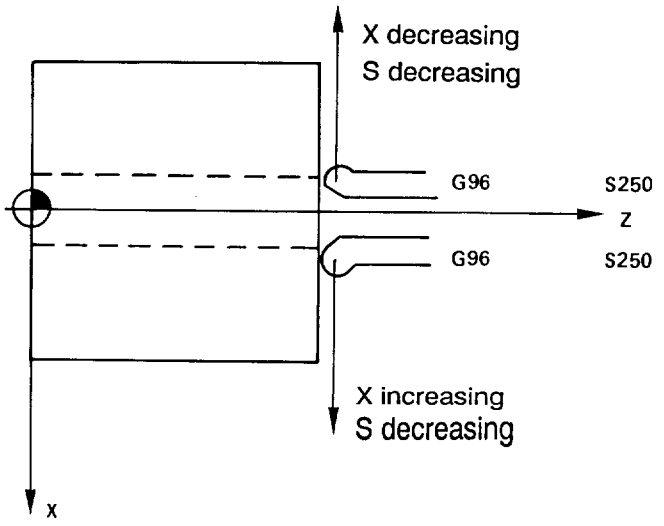
NOTES :

- *Whilst cutting with constant speed, it is possible to change the speed demand. However, the complete format, G96 S5, must be re-defined.*

- *Changing a tool compensator, (D), in CCSPD mode, will modify the cutting speed according to the difference in those values, which affect the workpiece diameter.*

- *An origin offset, (G54), will not affect the cutting speed.*

- *In the CCSPD definition block, (G96 X5.3 S5), the spindle speed is initially calculated with respect to the attachment dimension, X, and the demanded cutting speed, S. From this point on, spindle speed changes are a function of the distance and direction of movement along the X axis.*



- *Take care that the chosen speed range covers the required variations in spindle speed, because the system will automatically restrict itself to the maximum speed in the range.*

- *It is advisable to cancel CCSPD, (by G97 S5), before each tool change and to re-establish it later, with the new tool.*

7.3.3 – Spindle speed safety limit : G92

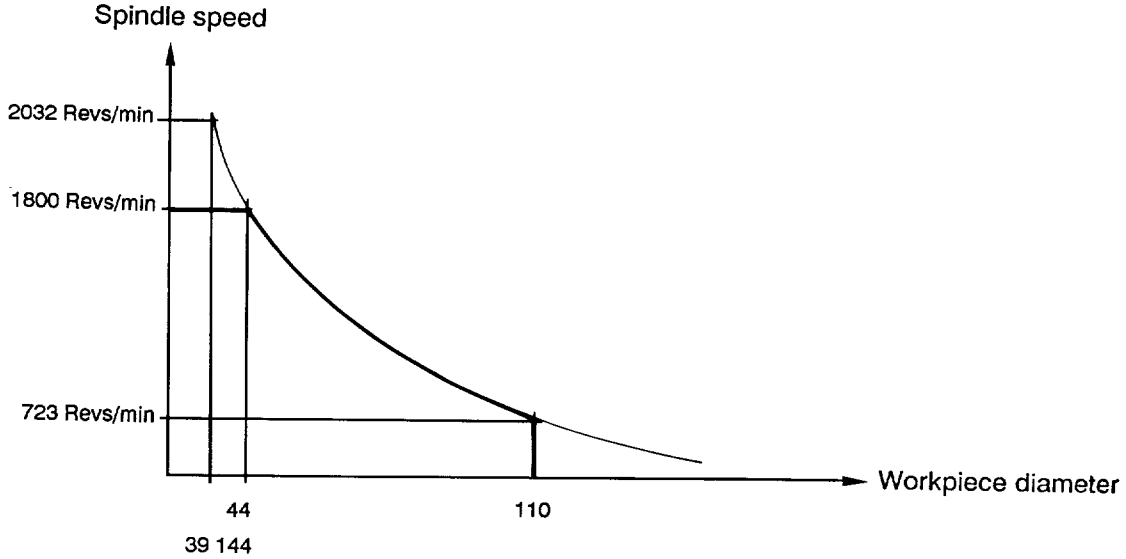
Format :

G92	S5
↓	↓
Spindle speed limit code	S sets the max. spindle speed in revs/min

It is possible to temporarily set a lower speed ceiling than that allowed in the range. This limit will not then be exceeded by constant cutting speed, (G96), as the diameter decreases. (This may be required by considerations of job or chuck imbalance, or of excessive centrifugal force, etc.).

To govern the maximum spindle speed, a G92 block should be programmed, before the G96 block.

In example 1 (see 7.3.2) an additional sequence, N95 G92 S1800, could be programmed, to set the ceiling at 1800 rpm. Then at block N100, the spindle would only rotate at 1800 rpm, and the speed would start to decrease only after the workpiece reached 44 mm diameter.



In the original example 1, (see 7.3.2), the spindle speed changed from 2032 to 723 rev/min.
With speed limit programming, (G92 S1800), the spindle speed would start at 1800 and drop to 723 rev/min.
The speed would not start decreasing until after passing the 44 mm diameter point.

NOTE :

- *In a G92 block, if an axis is programmed before S, the system will interpret the G92 function as a preset for the program origin on that axis, and not as a spindle speed limit.*

- *Spindle speed limits apply in all modes (CCSPD or not, feed in mm/rev. or mm/min.).*
  If any demanded speed is greater than the maximum value defined by G92, the spindle will limit its speed to this lower value.

7.4 – SPINDLE ORIENTATION

On those machines which incorporate this feature, M19 is used to orientate the spindle to any demanded angular position with respect to a fixed position as defined by the manufacturer. (See manufacturer's handbook).

The orientation position is given with the C address. The spindle cannot be orientated from a stationary condition.

EXAMPLE :

M3 S500 C90 M19

From an initial running condition of 500 rpm, orientate and stop the spindle, +90° from the origin.

If the C address is ommitted, the spindle will orientate to the origin.

8 - PARAMETERIC PROGRAMMING

Programmed parameters are functions which can be assigned to all addresses instead of numerical values. They can be used by the programmer as special functions.

There are two categories of parameters :

- program variables, L,
- program parameters, E.

8.1 - PROGRAM VARIABLES L

Two types of program variable are available :

- L0 to L19 variables,
- L100 to L199 and L900 to L939 variables.

These variables are identical in terms of format and use, but can differ in the way they affect the running of the workpiece program.

8.1.1 - Variables L0 to L19

These are initialized, (set to 0) when the system is switched on, or at the end of the workpiece program (M02), or by pressing the "RESET" button.

These variables can be a fixed value or they can be the product of the following operations;- addition (+), subtraction (-), multiplication (*), division (/), square root (R), sine (S), cosine (C), truncation (T), arctangent (A), logical AND (&), logical OR (!).

- Their use does not restrict program writing in any way.
- AND and OR operations are carried out on truncated values. (i.e. that part of the value which follows the decimal point is automatically removed. The original value is NOT rounded to the nearest whole number.)
- A maximum of 8 digits can be programmed, excluding the sign. The position of the decimal point is irrelevant.

L8 = 18
XL8 = X18 mm
FL8 = F18 mm/min } Assigning an L parameter to an NC address involves equating the unit of L with the unit of the corresponding address.

Program variables can be assigned to all programmable addresses, dimensions and functions. They can be used with G79, to carry out conditional program jumps.

Example of using a parameter

L2 = 5
L1 = L2 + 5.3 * 3 * S30 (Simple left to right execution gives, L1 = 15.45)
XL1 (The value of L1 is applied to the X dimension. ie X15.45 mm)

8.1.2 – Variables L100 to L199 and L900 to L939

These variables are identical in format and use to variables L0–L19. Using them, however, can have an effect on the running of the workpiece program.
Writing to one of the L100 to L199 variables will delay the preparation of that block until the PRECEDING block has been completed. (This is not so with the L0 to L19 variables.) This means that a block, whose execution takes into account the contents of the following block, (eg : geometric programming of the profile over 2 or 3 blocks, or tool radius compensation,) shouldn't immediately preceed a block, which either reads from or writes to, one of the L100 to L199 variables .

- However, by restricting access to PROGRAM EDIT, MDI modes and PLC sub–program calls, (via M999),

  the variables L100 to L199, and L900 to L939, may be used in the same way as variables L0 to L19.

  This M function prevents the operator, (or the PLC,) intervening in the progress of a series of blocks.

EXAMPLE :

```
%1                              %2
'                               '
'                               '
'                               '
'                               '
N90  X Y ...                    N80  M999
N100 L110 = 1+L110 ....         N90  X Y ...
N110 XL110 .....                N100 L110 = 1+L110 ...
N120 L120 = L110+L10            N110 XL110
N130 YL120                      N120 L120 = L110+L10
'                               N130 YL120
'                               N140 M998
'                               '
'                               '
```

In program 1, block 100 will not be prepared and carried out until N90 is finished.

In program 2, block 100 will be prepared before execution of N80, and there will be no "stop" at the end of block N90.

ADDITIONAL ASPECTS OF USING PROGRAM VARIABLES :

The variables L900 to L925 correspond respectively to the letters of the alphabet, A to Z, providing that they are immediately followed by an equals sign. For example, programming C=12.34, is the same as programming L908=12.34. This feature is most useful in developing custom macros.

- Function M999 :

  When the M999 function is programmed, the operator cannot call EDIT and MDI modes, nor can the PLC call a sub-program until the M998 function has been encountered. However, manual movements, (JOG or RECALL), the overspeed function, (G12), and the premature block termination function, (G10), are allowed during M999. When the system is switched on, or after a RESET or MO2, the initialized state is M998. (The M999 function is cancelled by M998 or M997.) Although it is possible to search to a block within a sequence covered by M999, it is not possible for the operator to enter search mode from a series of blocks "masked" by M999.

- Function M997 :

  The M997 function is used to force block continuation. If the system is in block by block mode when M997 is read, the following blocks will be run as if the system were in continuous mode until the program meets a cancelling function, M998 or M999 or M02 (refer to para. 9.10).

8.2 - PROGRAM PARAMETERS

Program parameters are defined by address E, followed by 5 digits. The most significant digit specifies the type of parameter.

8.2.1 - The different types of program parameters

Type 2

E200XX : Information on bits, written by the interface and read by NC

XX — Bit No., from 0 to 31

E21XXX : Bit information, read by NC : software option present (=1) or absent (=0)

XXX — Bit No. from 0 to 127

Type 3

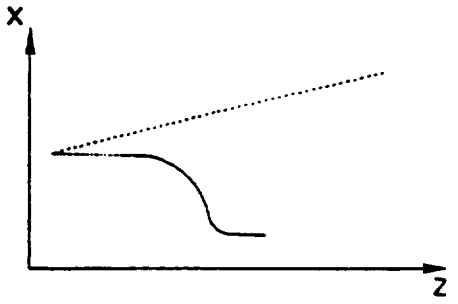
E31000
E31001 : These two read/write parameters are used to choose the type of graphic drawing for the part.
E31000 is relative to G0.
E31001 is relative to G1, G2 or G3.

The types of drawing are:
0 : solid line
1 : dotted line
2 : dashed line
3 : combined line
4 : stylus raised (no drawing)

These parameters are reset by a graphic reset of the system.
If a value less than 0 or greater than 4 is entered, error 92 is reported.

Example:

```
    %15
N10 E31000=4 G Z200 X150
N20 X50 Z150                          No drawing (stylus raised)
N30 E31001=0 G1 Z125 ES+ EB10         Drawing with solid line
N40 G3 X110 Z90 I75 K90 R35 ET
N50 E31001=1 G1 EA180 Z50             Drawing with dotted line
N60 E31001=2 G X150 Z200 M2           Drawing with dashed line
```



Type 4 : E40000 : Pure binary-coded data, on 32 bit word, written by the interface and read by NC. This parameter encompasses parameters E20000 to E20031

E41000 : Current mode. Read-only parameter

E41001 : Axis group number. Read-only parameter.

E41002 : Job reference. Read-only parameter

E41003 : Job reference. Read-only parameter
Graphic simulation status. Read-only parameter
= 0 no simulation
= 1 simulation with material removal
= 2 dynamic simulation (different from drawing during machining without axis movement).

Type 8 : E80XXX : Local data, written and read by NC

Data No. (from 0 to 49)

Type 9 : For Type 9 parameters the least-significant digit defines the order of the machine axis on a decimal base (0 or 1).

9000X : Axis measurement

9010X : Datum switch position in machine dimensions.
This parameter is the image of machine parameter P16.
Read-only parameter.
It can be used to create an automatic MOS program calculating the distance between the encoder zero pulse and the datum switch edge.

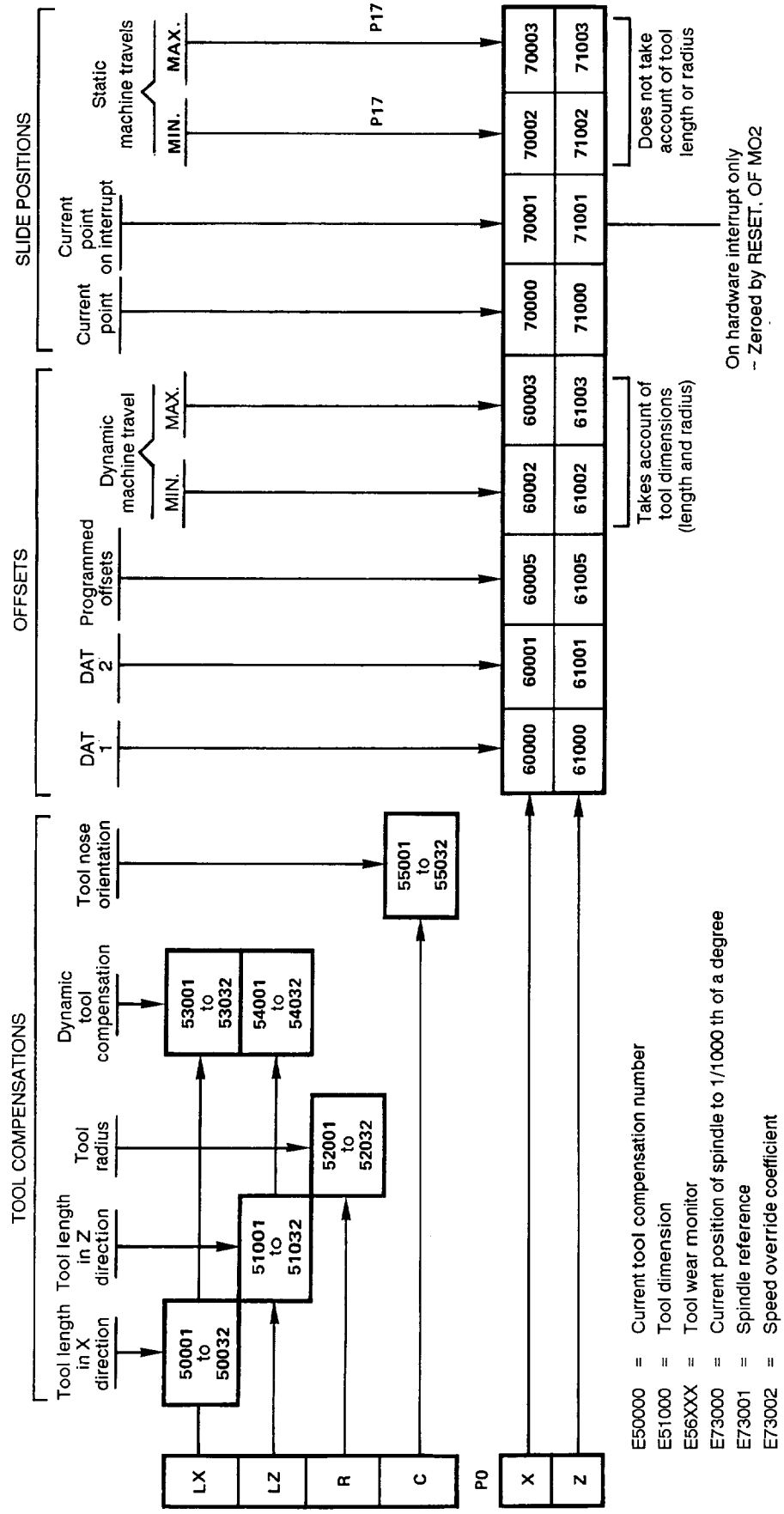
9100X : Status of the axes - servo-controlled or not

9200X : Validation or non-validation of m/c origin datum switches

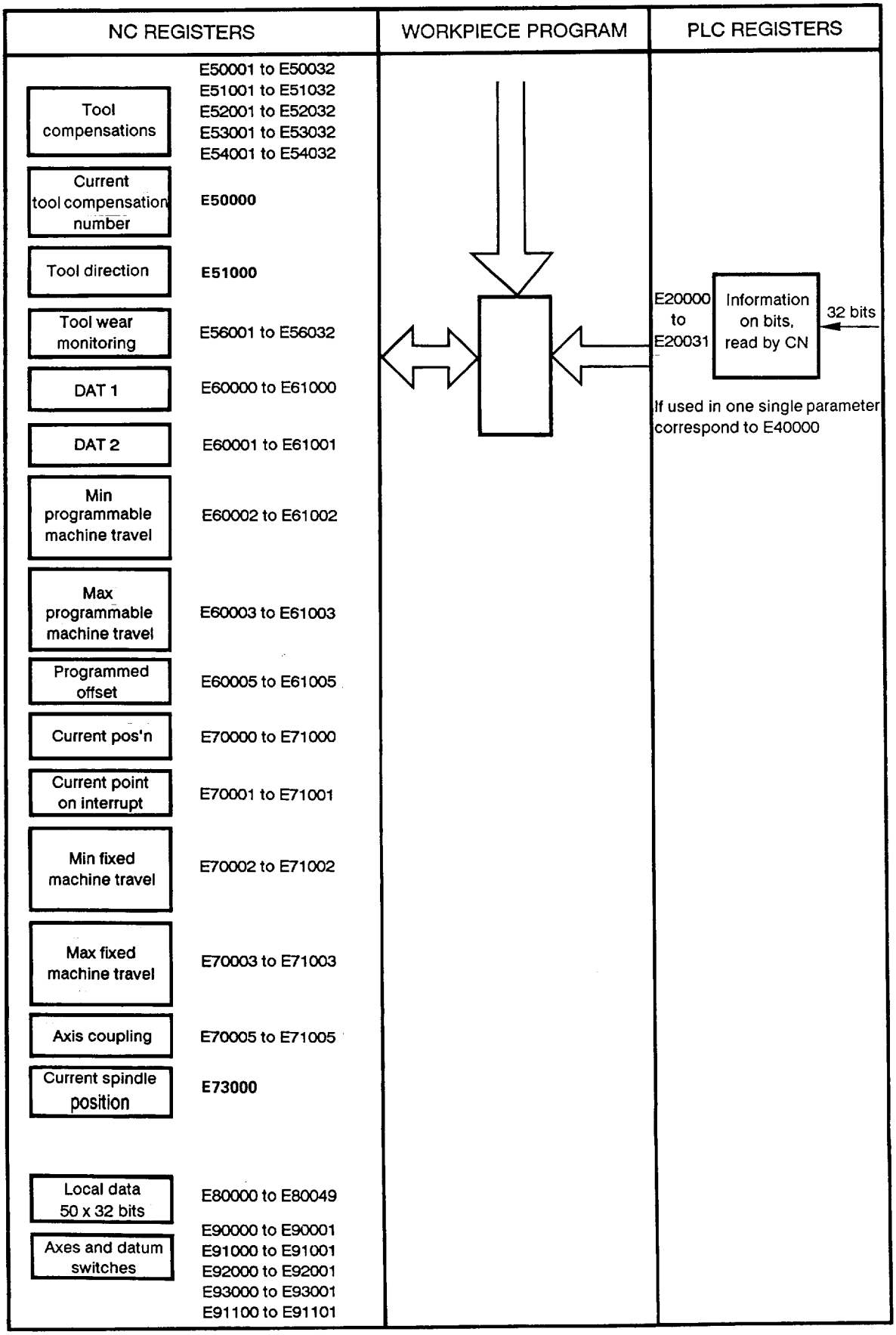
9300X : State of m/c origin datum switch

9110X : State of the MOS on an axis

NOTES


TOOL COMPENSATIONS
Tool length in X direction
Tool length in Z direction
Tool radius
Dynamic tool compensation
Tool nose orientation
50001 to 50032
51001 to 51032
52001 to 52032
53001 to 53032
54001 to 54032
55001 to 55032
LX
LZ
R
C
P0
X
Z
OFFSETS
DAT 1
DAT 2
Programmed offsets
Dynamic machine travel
MIN.
MAX.
SLIDE POSITIONS
Current point
Current point on interrupt
Static machine travels
MIN.
MAX.
P17
P17
60000
60001
60005
60002
60003
70000
70001
70002
70003
61000
61001
61005
61002
61003
71000
71001
71002
71003
Takes account of tool dimensions (length and radius)
Does not take account of tool length or radius
On hardware interrupt only
- Zeroed by RESET, OF MO2
E50000 = Current tool compensation number
E51000 = Tool dimension
E56XXX = Tool wear monitor
E73000 = Current position of spindle to 1/1000 th of a degree
E73001 = Spindle reference
E73002 = Speed override coefficient
E73003 = Programming of minimum block execution time
E7X004 = Shift direction
E7X005 = Assigning of axes

8.2.2 – Summary of program parameters



8.2.3 – Operating precautions

The program parameters are never reset to zero by the system.

Unlike the program variables, which are linked solely to the workpiece programs, the use of program parameters is subject to several restrictions :

- Type 7 parameters are "read only".
- Any operation, (read or write,) carried out on an program parameter stops preparation of the program until the end of the preceding block. Therefore, a block containing an program parameter cannot be preceded by a block whose execution requires "knowledge" of the contents of the following block or blocks (geometric programming, radius compensation, etc ...)
- The value of an program parameter is ALWAYS an integer. (Unlike the program variables, L)
- When an external parameter is assigned to an NC address, it equates the parameter unit with the decimal part of the corresponding function :

  If E80000 = 18000
  XE80000 ---> X18.000 ---> X = 18 mm
  FE80000 ---> F180.00 ---> F = 180 mm/min

- An E parameter can itself be parameterized.
  If L0 = 80003, then EL0 corresponds to E80003, so EL0 = 234.567 will put a value of 235 into E80003. (Remember that the system will put an integer value into the E parameter, and will automatically round up or down to the nearest whole number.)

8.2.4 – Using type 9 program parameters

Remember that an operation on one of these parameters stops movement at the end of the block preceding the block in which it is programmed.

8.2.4.1 – Axis measurement : E900xx

The absolute position of all axes, (with respect to m/c origin,) can be read, irrespective of whether the axes are measuring only or fully servo-controlled. However, only axes which are measuring only, and not servo-controlled, can be written to.

8.2.4.2 – Status of the axes : servo-controlled or not servo-controlled : E910xx

This parameter is used to modify the axis status as originally defined by machine parameter P3, (P3 retains its original value). The status of all machine axes can be read. In write mode, all the axes can be assigned servo-controlled or not servo-controlled.
When the value of the parameter is "1", the axis is servo-controlled. "0" means that it is not servo-controlled.

NOTE :

When the system is re-initialized from power up or reset or M02, the initial status of the axes, as defined by P3, is restored.

8.2.4.3 – Recognising the m/c origin datum switches : E920xx

This command status can be read for all axes of the machine which have position feedback.
This parameter can be written to, if the axis is "measuring only", (ie NOT servo–controlled).
When the value of the parameter is 1, the datum switch is enabled, (0 = disabled).

NOTE :

Disabling is automatic on contacting the reference switch, (during m/c datum setting), or when a reset or an M02 occurs. In m/c referencing mode, it takes a value of 1, as soon as the axis jog pushbutton is pressed, and changes to 0 when the pushbutton is released, or when the datum switch is met.

8.2.4.4 – Origin datum switch status : E930xx

This is a read only parameter, which is accessible on all the measured axes of the machine.

A value of "1" indicates that the datum switch is currently made, "0" indicates that it is not.

8.2.4.5 – Acknowledgement of the MOS on an axis : E911XX

This parameter is used to identify an axis which has been previously referenced.

If the value of the parameter is zero, the axis has been referenced. If the value is 1, the MOS has not been set.

This is a read/write parameter. (If an axis movement is demanded of an axis which has not previously been referenced, ie E911xx is a 1, the system will report an error.)

8.2.5 – Special type 5 parameters

- E50000 : Current tool compensation number. (Dxx). Read only parameter.

- E51000 : Tool direction. This parameter gives the physical address of the axis parallel to the direction of the tool. If the tool direction is negative, the value of the axis address is increased by 100. This is a read only parameter.

  EG : E51000 = 100 (negative direction on X axis)
  E51000 = 0 (positive direction on X axis)

- E560XX : additional element (H) in the table of dynamic tool compensations, which can be used to monitor tool wear or other data (max. 8 figures). This is a read/write parameter.

  The dynamic tool compensation page is displayed as :

D1	DX + 0.123	DZ + 0	H + 12345678
D2	DX + 12.456	DZ + 1.325	H + 1200

  The format of DX and DZ is 2.3, with a maximum of 99.999.

  The values of H may be entered manually. The appropriate syntax is : Dxx Hxxx (LF).

  This table can be input or output by tape or via the DNC 1 link.

8.2.6 – Special type 7 parameters

- E7n005 : assigning axes.

  This parameter is the part program equivalent of machine parameter P9. It is used to transfer a program axis from one machine address to another.

- E7n004 : shift direction, n program axis.

  In linear interpolation, this parameter contains the relative dimension along axis n of the current block.

  In circular interpolation, it contains the relative component along axis n of the current radius offset by + or – 90 degrees depending on whether function G3 or G2 is in use. This parameter is read only.

- eg. E7n005 = m where n = program axis, (P0 order). m = machine axis, (P9 order).

  If E7n005 = –1, the program axis (n) is uncoupled. ie freed.

  A program axis MUST be free, before being assigned to another machine axis.

  This is a read/write parameter.

  The original axes assigned by parameter P9 will be restored following a reset or M2 .

8.2.7 – Using parameters E41000 and E41001

When a workpiece program is run in "test" mode, the following functions are inoperative :

- datum switch stop,
- priority interrupts,
- sub–program calls by the PLC.

Similarly, in conversational graphics mode, the following operations are not carried out :

- writing to any program "E" parameters,
- sub–program calls, by M function,
- message transmissions to the screen, via $0.

As a result, if the program contains conditional return jumps, the program may unintentionally be sent into a permanent loop.

Such problems may be resolved by testing parameters E41000 and E41001.

- E41000 : current mode. (Read only parameter.)
  If the system is in test mode, E41000 = 6. In search mode, E41000 = 4, etc.
- E41001 : axes group number. (Read only parameter.) This parameter is equal to 1 in graphic mode, otherwise its value is zero.

8.3 – OPERATIONS ON PARAMETERS

List of operators

Symbol	Operation	Links together the various terms in an expression (1)	Qualifies the following term
+	Addition	yes	yes
–	Subtraction	yes	yes
*	Multiplication	yes	no
/	Division	yes	no
R	Square root	no	yes
S	Sine (2)	no	yes
C	Cosine (2)	no	yes
T	Truncation (3)	no	yes
A	Arctangent (4)	no	yes
&	AND (5)	yes	no
!	OR (5)	yes	no

(1) The operation binds the term following the symbol into the result of the preceding operations (no priority, simply left to right).
(2) The following term is expressed in degrees, or 1/1000ths degree, if an E parameter.
(3) Forms an integer value by removing the decimal part of the term following the symbol.
(4) The result of the operation is expressed in 1/1000ths of a degree.
(5) These logical operations are carried out on automatically truncated values. (i.e. that part of the value which follows the decimal point is removed).

– Conditional jumps

General case :– G79 (Parameter), (Test), (Expression), N...

The system will jump to block N... if the test is "true", otherwise it will proceed to the next block.

Comparison symbols : = equal
> greater than
< less than

Any TWO symbols may be used together, to create any particular test.

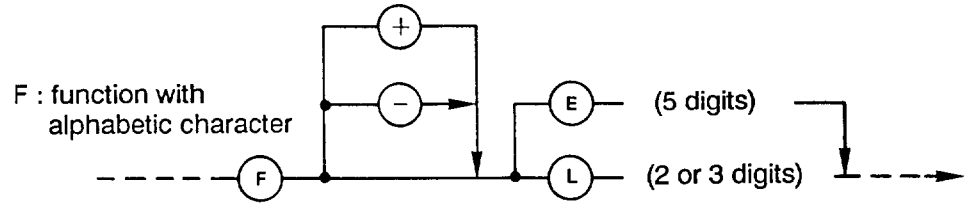
NOTE :

Programming : G79 N... (without a test) will produce an unconditional jump.

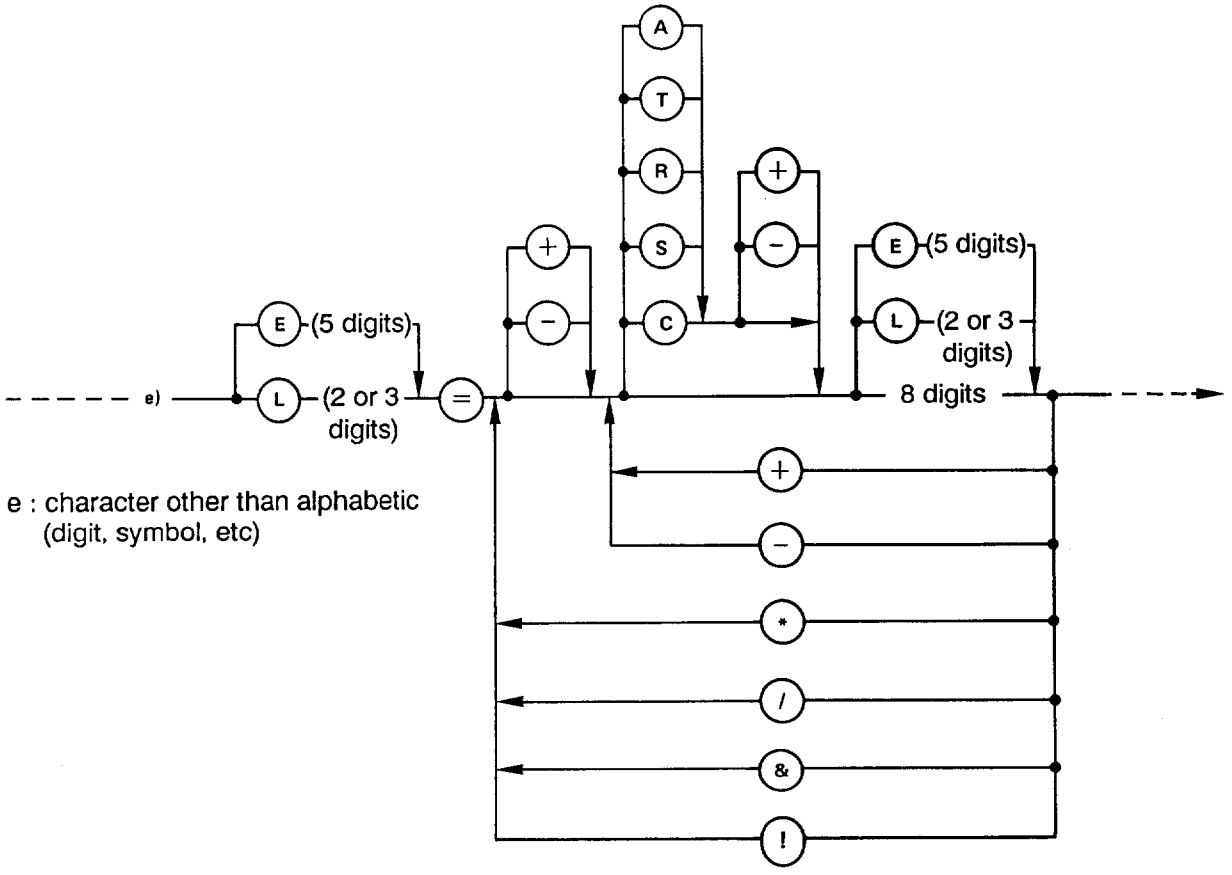
8.4 – SYNTAX OF PARAMETERIC PROGRAMMING

L (2 or 3 digits) : program variable.
E (5 digits) : program parameter.

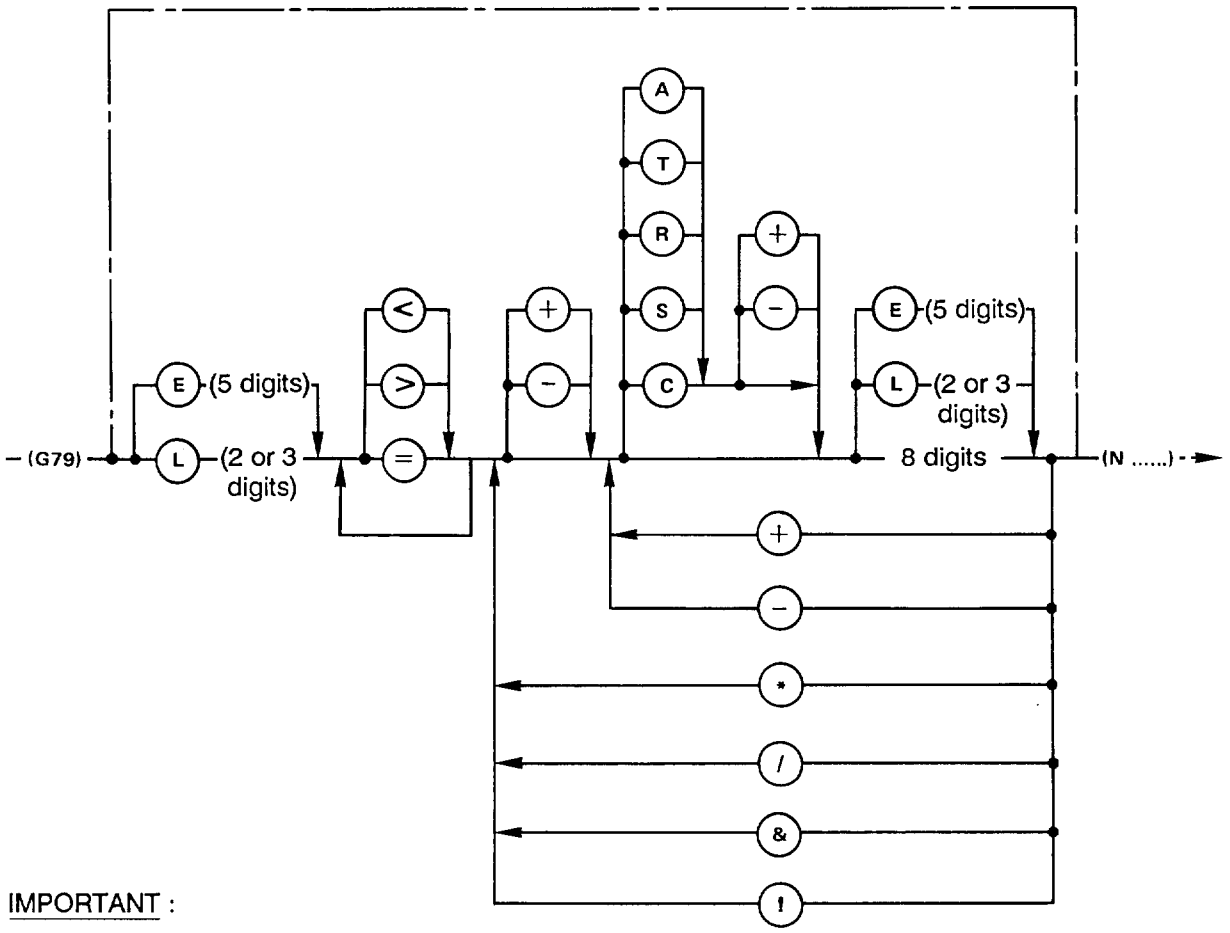
8.4.1 – Assigning a parameter to a function



8.4.2 – Declaring a parameter



8.4.3 – Testing a parameter (comparison with an expression for a conditional jump)



IMPORTANT :

When fractional values are used to increment program variables (L), the results of a test may be unexpected, due to rounding, even though the displayed value of the parameter may look correct. Take care over this, particularly when planning a loop.

EG :

```
L1=0
N10 .....
'
'
'
L1=L1+0.6 G79 L1<6 N10
```

After 10 loops, L1 shows 6 on the display. The true result of the computation, however, is slightly less (perhaps 5.9999994). This is because 0.6, held as the computer holds the number, (in binary,) cannot be completely exact. Thus, an additional unexpected pass through the loop may be made.
If it is absolutely necessary to increment in fractional values, make the test on a deliberately "safe" value :

Eg. for 0.6 with 10 loops, test for 5.9
for 0.01 with 10 loops, test for 0.096, etc.

8.5 - EXAMPLES OF USING E PARAMETERS

8.5.1 - Operations with E parameters : Division

The result of the division of an E parameter by another E parameter can only be fully stored in an L program variable, because, as E parameters have no decimal part, the results stored in them can only be whole number values.

EXAMPLE :

E80002 = 3150
E80016 = 2400
L1 = E80002/E80016 (= 1.3125)

If E80005 had been written instead of L1, the stored value would have been 1.

If the result has to be transferred to an E parameter, it will be rounded to the nearest whole number.

If, in the example above, only 3 digits are to be retained after the decimal point, one could write :

E80005 = L1*1000 Then, E80005 will contain 1313

The following could also have been written :

E80005 = E80002/E80016*1000. The result would still be 1313

8.5.2 - Using parameter E7(n)001

These parameters are used in conjunction with G10, to save all of the slide positions following an NC priority interrupt, (eg. from a probe or a datum switch stop). The stored values are retained in the E7n001 parameters until a new G10 is encountered, or until M02 or reset.

When a G10 is read, the parameters are automatically pre-loaded with a value of 99999999. This allows a test to be carried out to determine if an interrupt occured or not after the G10 block.

EXAMPLE : Modification of DAT1 Z (E61000) by probe sensing

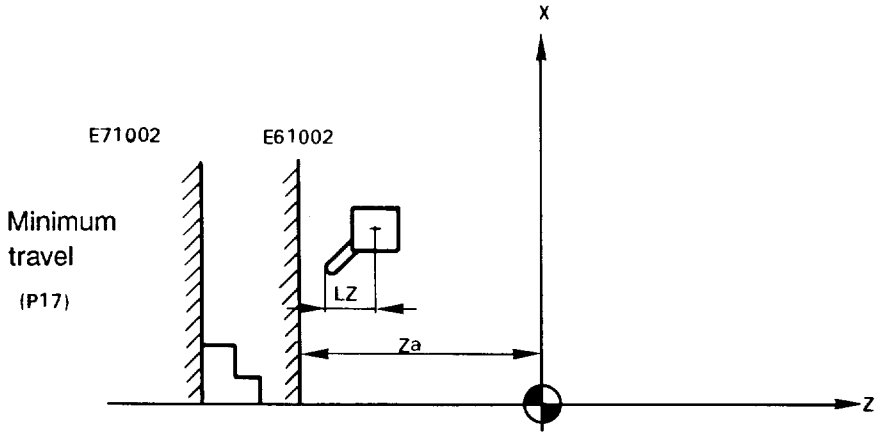
G X0 Z0	
G Z85	Rapid approach
G1 G10 Z100 F500	Probing move towards Z100 at 500mm/mn
G79 E71001>90000000 N100	Test whether the interrupt occured or not. (Jump to N100 if it did not).
E61000=E71001	Load DAT1 Z, with the Z interrupt value.
N100 Continuation of the program	Depending on the test, the program continues with the old DAT1 value, or with a new one.

8.5.3 – Using parameters E6(n)002 and E6(n)003

In normal operation, the system checks all axes demands against software limits. These limits, (contained in P17), are pre–set by the m/c builder, and define the maximum working envelope.

To prevent collisions, these limits may be temporarily modified to suit the conditions of a particular workpiece to be machined. To do this, the E6(n)002 and E6(n)003 parameters are used to re–define the limits, but always within those defined by P17.

When the system is switched on, or when there is a RESET or M02, these parameters are initialized to a value of +/– 90000000, (depending on whether the parameter governs the maximum or minimum travel).



To modify the limits, the programmer should write :

E61002 = Za

Whilst machining, the system takes account of the tool compensations so as not to exceed the defined limits. Thus, in maximum X travel, the axis cannot exceed Za – LZ.

8.5.4 – Using type 9 parameters and axis equivalence

Automatic machine origin setting on two axes : X (axis 0) and Z (axis 1)

Before starting the program, the MOS must be declared via the keypad

```
%7001
G79 N100
N10                                  @X : axis name, L0 : axis number
L6 = 91100+L0                        MOS recognised
L1 = 90000+L0                        Measurement value
L2 = 91000+L0                        Servo-controlled/not servo-controlled
L3 = 92000+L0                        Datum switch enabling
L4 = 93000+L0                        Datum switch status
EL6=0                                Declaration of MOS done on an axis
EL2=0 EL1=-1000 EL2=1 L5=EL1/1000    Initialize measurement to -1.
N11 G79 EL4=0 N12                    Jump to N12 if not on the datum switch.
G52 G0 L5=L5-1 @XL5 G79 N11          Else, move off the switch in 1mm steps, (depending on MOS
                                     direction)

N12 EL2=0 EL1=-50000000 EL3=1 EL2=1  Initialize measurement at 50m, the sign depends on the
                                     direction of the MOS

N50 G52 G1 G10 @X0 @L0>0             Move towards the measurement origin till the switch is
                                     detected.

N100   @X=X L0=0 G77 N10 N50
       @X=Z L0=1 G77 N10 N50         Definition of the axes equivalences
```

8.6 - UPGRADES IN PARAMETRIC PROGRAMMING

These upgrades require an NC software at index D or above.

8.6.1 - Read of the Modal Functions of the Current Block

The symbols for these functions preceded by the character "." (dot) are declared between square brackets. These functions are accessible by parametric programming, but for read only.

- Boolean functions : G and M functions

  The symbol for these functions is their number preceded by the character "B".

  Example : [.BG01] ; [.BG70] ; [.BM05]

  . List of G functions :

  [.BG00] [.BG01] [.BG02] [.BG03] [.BG17] [.BG18] [.BG19] [.BG23]
  [.BG90] [.BG91] [.BG40] [.BG41] [.BG42]
  [.BG93] [.BG94] [.BG95] [.BG96] [.BG97]
  [.BG80] [.BG81] [.BG82] [.BG83] [.BG84] [.BG85] [.BG86]
  [.BG87] [.BG88] [.BG89] [.BG70] [.BG71] [.BG74]

  . List of M functions : (decoded)

  [.BM03] [.BM04] [.BM05] [.BM07] [.BM08] [.BM09] [.BM10] [.BM11]
  [.BM19] [.BM48] [.BM49] [.BM997] [.BM998] [.BM999]
  [.BM40] [.BM41] [.BM42]

- Auxiliary functions of encoded values :

  . [.RF] Feed rate - Format 5.2 in mm/min

  . [.RT] Tool No. - Format 5.0

  . [.RS] Spindle speed

  Example (for the type of programming, see Chapter 12).

- Programming in metric units, then in inches, followed a return to metric units.

```
% 3
NI G0 X Z <--  positioning in metric units if the system is initialised in metric units (G71).
               (G71).
VAR [INC MIC]
ENDV
[INC MIC] = 71* [.BG71] + [INC MIC]
N10 G70 X12 Z32      )
'                    )
'                    ) Programming in inches
G [INC MIC] X5 Z2    )
'                    )
'                    ) Programming in metric units
M2                   )
```

- Spindle indexing sub-program.

```
% 8000

L0 = [.BM19]
G79 L0 = 1 N2     Jump unless already indexed
L1 = [.BM05]
G79 L1 = 0 N1     Jump unless spindle already rotating
S100 M3           Start spindle rotation
N1 M19            Index
N2...             Start of the operation requiring spindle indexing.
```

8.6.2 - Acquisition of a Value Entered by the Operator from the Alphanumeric Keyboard

The value entered by the operator can be one of the terms on the right-hand side of a parametric expression.

- This value is symbolised in the expression by the character $.

  e.g. : L0 = L1 + $

- The value entered by the operator can be a number with a maximum of 8 digits plus sign digit and decimal point. This value is confirmed by pressing LF.

  When it reads $ in the program, the system stops and waits for the operator to enter a value.

- When the first character entered by the operator is alphabetic, the system encodes this character as a value between 1 and 26 (A = 1, B = 2, ..., Z = 26) and this value is immediately confirmed.

- The characters entered are echoed on the CURRENT, CURRENT PT and DYNAMIC TRACE pages as messages programmed after $0 or additions to a message already programmed.

- The only active function key besides the LF key is the <== (back space) key used to delete characters entered before they have been confirmed.

8.6.3 – Display of a Value After a Programmed Message

When the symbol $ is the destination operand of a parametric expression, the value resulting from this expression is displayed after the message already programmed after $0.

The character = after the symbol $ assigns the result of the expression to the display.

e.g. : FOR L0 = 1 T0 20 D0 $ PHASE N0 : (LF) (For this type of expression, see Chapter 12).
$ = L0
.
. processing
.
ENDF

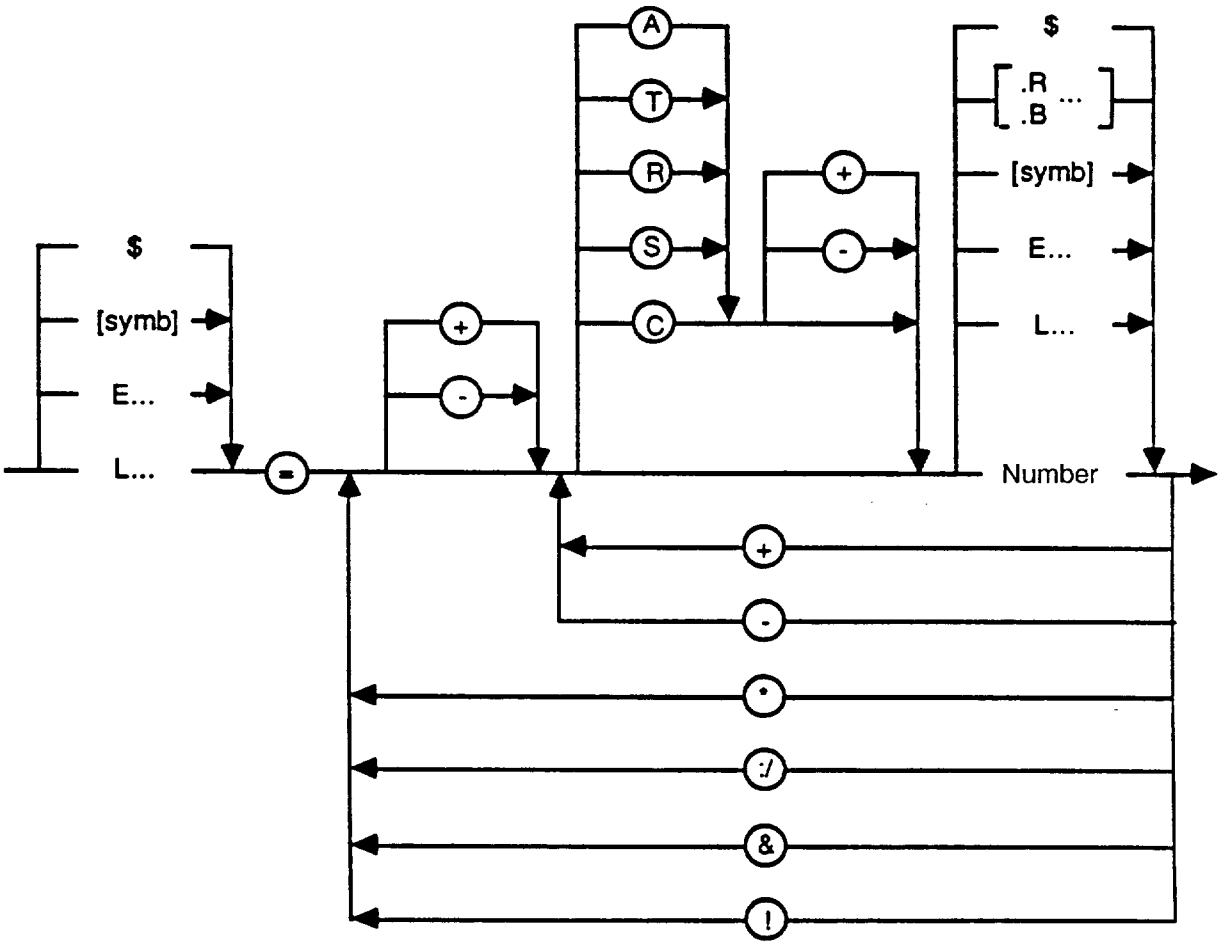
When the system reads the sequence $ = L0, the message PHASE N0 : (value of L0) is displayed.

CAUTION :

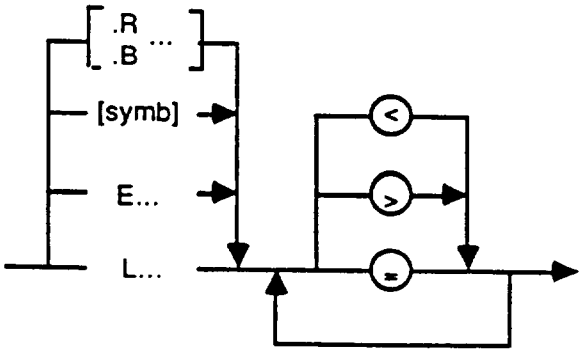
The function $ = ... located in a block after function G79 is not a jump condition.

8.6.4 - New Syntax Graph for Parametric Programming

- Load of a parametric expression :



- Comparison for conditional jump



(right-hand side of the expression same as the load expression)

NOTES

9 – MISCELLANEOUS FUNCTIONS

9.1 – OPTIONAL BLOCK "/"

Blocks preceded by a slash code, (eg. /N...) can be omitted at the operators' discretion, by pressing the "optional block" key.

The key–press operation is only effective in an M00 or M01 or M02 state.

9.2 – PREMATURE BLOCK TERMINATION

9.2.1 – Premature termination, via an external interrupt

If G10 is programmed in a movement block, the system is primed to expect an interrupt, which, if it occurs, will prematurely teminate that move and save, (for later analysis), ALL the axis positions at the moment of the interruption. Such interrupts are usually the result of a probe input. If the expected interruption occurs, a jump may be made to a designated block.

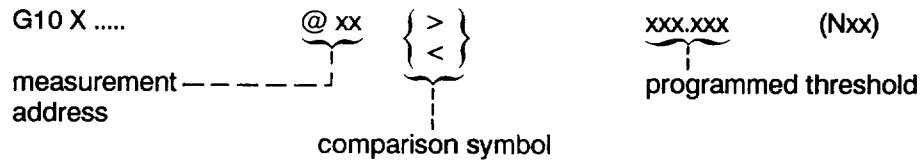
The G10 function is only allowed in G40 mode. Axis movements can be executed in G0, G1, G2 or G3. Dwells, in G4, can also be interrupted.

eg. G10 X12.345 Nxx. If Nxx is ommited, (or if the interrupt fails to occur,) the following block is executed.

9.2.2 – Premature termination, via measured threshold detection

As well as probe type interrupts, the system can be made to generate its own interrupt by comparing the measurement of an axis to a programmed threshold.

The block syntax is :



- The measurement address is that axis number, connected to the encoder feedback, as recognised by the system via the m/c parameter P9. If Z is to be measured and the axis number of Z is 1, @01 will be programmed. All axes of the system can be used, whether they are measuring only or fully servo–controlled.

- Comparison symbol : defines the interrupt condition.
  ≥ greater than or equal to
  ≤ less than or equal to

  The designated axis is compared to the programmed threshold every 10 ms. When the condition is true, the block is interrupted.

- Programmed threshold : a signed value with a decimal point expressed in the current program units.

- Example :
  G10 X ... Z ... @ 0 < 50.32 N100

  Jump to sequence 100 when the measurement of axis 0 is less than or equal to 50.32.

9.3 – SUB–PROGRAM CALLS AND PROGRAM BRANCHING

9.3.1 – Sub–program calls with return : G77

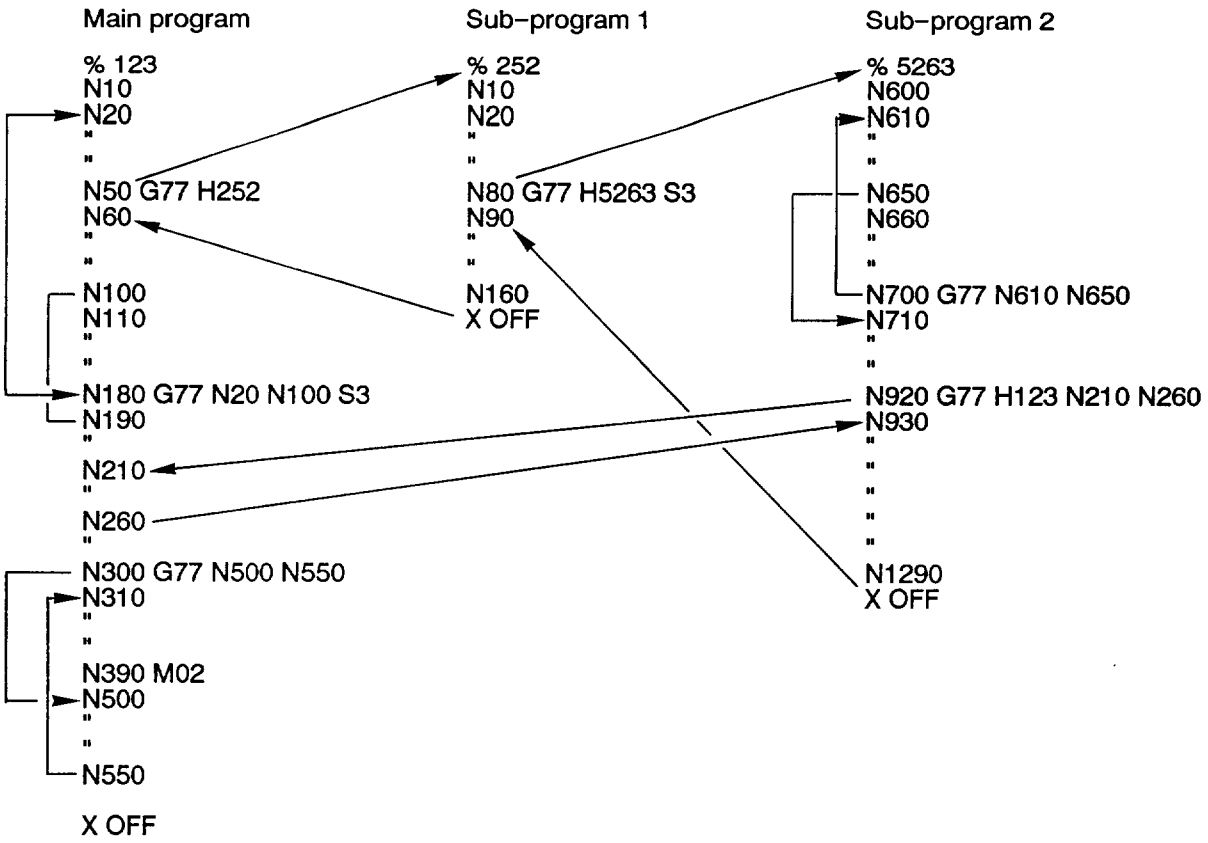
The call is unconditional.

The sub–program (or machining macro) can be defined in the main program. Each sub–program can be repeated up to 9999 times (S4), and upto four levels of sub–program nestings are possible.

An INTERNAL sub–program is defined within the main program, and is identified by its start and end block numbers. It can be situated between the % code and the M02 or even beyond the M02, but before the XOFF character. Any series of blocks in a program may be defined as a sub–program.

An EXTERNAL sub–program is identified by the letter address H, followed by its program number, optionally followed by start and end block numbers.

EXAMPLE :



- Sub-program 5263 and N20/N100 from program 123 are run 3 times (S3).
- Sub-programs 1 and 2 must not include function M02.

A sub-program defined by two sequence numbers can only be executed in the order of execution of the program.

EXAMPLE :

These two formats are identical.

NOTE :

If a program in the workpiece program area of RAM memory has the same number as a resident sub-program, (in non-volatile EEPROM), the RAM version will always take priority and will, effectively prevent access to the resident sub-program.

9.3.2 – Block jump, without a return : G79

A block jump will go directly to a designated block. The block called must be in the same program.

- Unconditional jump

  eg. G79 N280 : jump to block N280, which can either be before or after the current block.
  Caution : beware of creating an endless loop with backward jumps.

- Conditional jump

  Use of a program variable or a program parameter in the jump block, and on the condition
  > either = or<, or two simultaneous conditions.

  eg. G79 L2>=3 N280 : if L2 is greater than or equal to 3, jump to block N280, otherwise continue to the following block.

  (See paragraph 8.3 for more details of possible tests).

9.4 – EMERGENCY WITHDRAWAL : G75

The G75 function must be followed by a sequence number. This sequence number defines the beginning of the withdrawal program. The withdrawal program ends with a programmed stop M00 or an end of program M02.

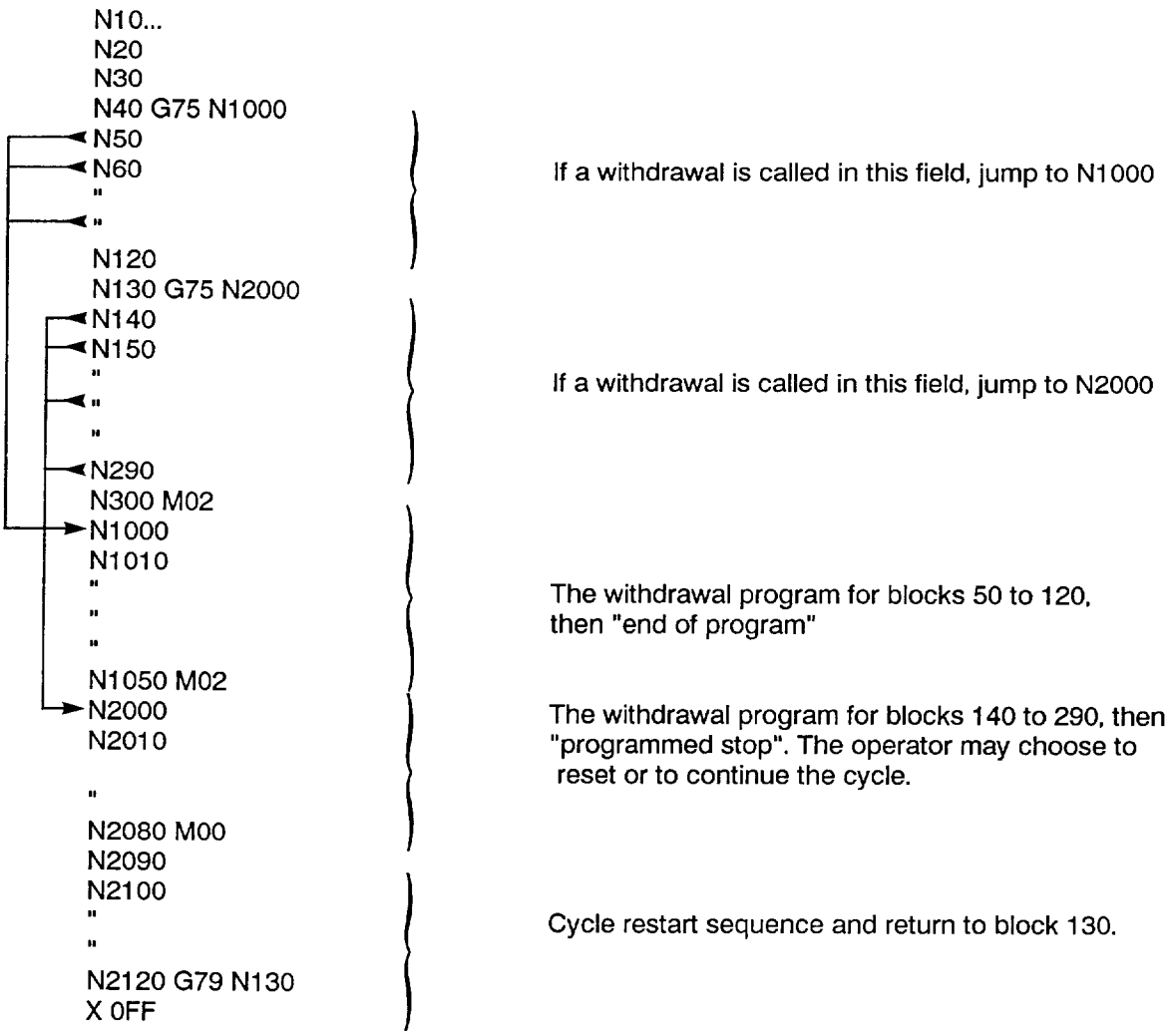
A withdrawal program definition is operative from the time it is encountered until it is superceded by a new definition from another G75.

An emergency withdrawal demand, is transmitted by the PLC, via the interface signal DEBURG. When the status of this signal goes to 1 and the system is in CONT, SINGLE BLOCK, MDI or RAPID mode, the block in progress is interrupted, in mid-block, and the system jumps to the last declared withdrawal program.

Whatever the mode in use, the withdrawal program runs in CONTINUOUS mode.

If a withdrawal program terminates at a programmed stop M00, this M00 must be followed by an operator restart.

EXAMPLE :



NOTE :

- *If an emergency withdrawal is demanded when no withdrawal program has been defined, the signal has the same effect as the FEED STOP button.*
- *If G75 N0 is programmed, an emergency withdrawal requested in the following blocks will result in a feed stop.*

9.5 - THE SUB-PROGRAM CALL BY THE PLC

The PLC can call a specific sub-program with the number %9999.

To call this sub-program, the PLC sets the APP SSP data in the interface to the NC.
If %9999 has not been created, Error 25 will be reported.

9.5.1 - Pre-conditions for call acknowledgement

The system must be in CONT, SEQ or RAPID mode.

When a workpiece program is being run, the call cannot be executed until the END of an interruptible block. A non interruptible block is a block internally created by the system during a fixed machining cycle (eg. G65, G66 ...), or a block which uses values in the following blocks for its processing. (eg. G41, G42 or GPP.)

While the sub-program is being run, further PLC demands for that same sub-program (ie. the state of APP SSP,) will be ignored.

As no direct acknowledgement is sent by the system to the PLC, it is usual for the sub-program itself to send an entry acknowledgement to the PLC via an M code.

9.5.2 - Sub-program entry

Modal data in the last block executed prior to the %9999 call, is saved.

Functions G01 and G40 are forced, on entry into %9999.

9.5.3. - Sub-program return

The modal state of G40, G41 or G42, which was active before the call, is restored, as are the modal functions (M.. and S..) and the program variables (L0 to L19). The compensator number (D...) is restored, only if the tool has not been changed. However, the variables L100 to L199 are not restored.

The position of the axes before the %9999 sub-program call will be automatically re-established on exit. The type of move, (G0 or G1,) to re-establish those positions must be defined on leaving the sub-program.

If the system was in G41 or G42 status before the %9999 call, repositioning will be made normal to the next block of the workpiece program.

All modal G codes are restored and the workpiece program continues.

NOTE :

If there was no workpiece program in progress when %9999 was called, the cycle ends on completion of the sub-program.

9.5.4 - Structure of the sub-program

When several functions can be processed by the sub-program, the PLC must specify which action is required. It can do this through, for example, parameter program E40000.

- 1st Method

  Each function is written in a different sub-program (%a, %b, %c, etc.). In this case, sub-program %9999 consists of a single block which acts simply to re-direct the actual execution.

  Example : %9999
  G77 H E40000 Mxx

  E40000 contains the number of the sub-program demanded (a, b, c, ...), function M serving as an acknowledgement.

  This method has the disadvantage of creating an overlap with the additional sub-program.

- 2nd Method :

  All the functions are written in sub-program %9999, whose first block is a jump to a sequence number contained in parameter E40000.

  Example :
  ```
  %9999
  G79 N E40000 Mxx
  Na  {processing 1st function
      {
  G79 Nz
  Nb  {processing 2nd function
      {
  G79 Nz
  Nc  {processing 3rd function
      {
  Nz End of sub-program
  X OFF
  ```

NOTE :

The sub-program normally ends with X OFF. If it is required to end the cycle in progress at the end of the %9999 sub-program, an M02 can be programmed.

9.6 - DISPLAYING A MESSAGE ON THE SCREEN

Strings of characters, (eg. messages or commands relating to the machining process, which are to be transmitted to the display screen,) can be included in the program, but will not interpreted in any way by the system.

These messages begin with the characters, "$0" and end with the LF character. (The zero is optional.)

eg. N100 G X Y $0 ...message text... LF

Messages transferred to the screen can be read on the CURRENT MACHINE POSITION or IN PROGRESS display pages.

Only one message at a time may be displayed. A new message erases the previous one. Should a programming error be detected by the system, while a $0 message is being displayed, the error number will be displayed on the second line.

The number of characters in a message should not exceed 39. If the message is longer, only the first 39 characters will be displayed. No error will be reported.

$0 LF, or an M02 or a RESET will erase a displayed message.

It is possible to add to an existing message to be displayed.

The syntax is : $ + added message

EXAMPLE :

```
$axe X =                    X axis = 123.456          Z axis = 654.321
$ = E90000/1000             where   E90000 = 123456
$+ axe Z =                  and     E90001 = 654321
$ = E90001/1000
```

9.7 - PROGRAMMING IN INCHES : G70

This option allows dimensions to be entered directly in inches.

Inch programming is selected with the G70 code. This function is modal and is cancelled by G71 (metric programming).

It is possible to configure the system to default to either imperial or metric at switch on or RESET, by the appropriate setting of bit 3 in machine parameter P7.

- in Inch mode, the dimension and feed rate format is as follows :

  X, Z, I, K, +044
  P, Q, R, EB

  F in G94 043
  F in G95 014

- The format of the other addresses do not change.
  The format of offsets and tool dimensions is as follows :

  DAT 1, DAT 2 +044

  Tool dimensions L 034
  R 024

Program variables (L) and program parameters (E) are simply numerical values. The particular use to which they are put, will dictate the units applied.

```
G70
L1=10
XL1                     (X = 10 inches)
E80000=100000
XE80000                 (X = 10 inches)
L10=E50001/25400        (tool length offset 1, converted to inches).
L2=100+L10              (Internally the values are saved in microns).
XL2                     (X = 100 inches + tool length offset 1 in inches).
```

It is the programmers' responsibility to adapt operations on dimensions and E program parameters, to match the appropriate units, as required.

9.8 - SUB-PROGRAM CALLS VIA M FUNCTIONS

A sub-program may be called directly by an M function.

Upto eight different M functions may be pre-defined to call a sub-program. Each M function/sub-program pair is entered, by the m/c builder, in machine parameter P35.

For example, if machine parameter P35 defines that M55 corresponds to sub-program %200, then, Nxx M55 in the workpiece program will act as Nxx G77 H200.

The return conditions are the same as for G77.

CAUTION :

Take care with the S address when using sub-programs called via M codes. The results obtained will vary according to the position of S, in relation to the M code.

Continuing with the last example :

Nxx S1000 M55 will demand a spindle speed of 1000 revs/min and will call %200, the sub-program defined by M55, once.

Nxx M55 S1000 corresponds to 1000 consecutive calls to sub-program %200.

NOTE :

In a sub-program called by an M function or by the PLC, it is not possible to call another sub-program using an M function or the PLC. (It is possible to use a direct G77 sub-program call.)

9.9 - AUTOMATIC COMPUTATION OF CUTTING TIME

When a workpiece program is stored in the memory, the sum of all the individual axis movement times may be computed. This indication of time, takes no account of m/c synthetics. (ie. dwells caused by machine actions, such as turret indexing, or spindle accelleration, etc.)

The overall time is calculated by running the program in TEST mode. The result is expressed in minutes and seconds (max. 546 minutes). It can be viewed on the program LIST page.

eg. %1 6532 BYTES 86' 37"

9.10 - "FORCED BLOCK CONTINUATION" MODE : FUNCTION M997

The blocks programmed after this function are executed continuously until the program meets M998, M999 or M02.

For example, if the system is in block by block mode when M997 is read, the following blocks will be run as if the system was in continuous mode until the program meets a concelling function. Similarly if a sub-program is called by an M function in MDI mode and M997 is programmed, the sub-program will be run in continuous mode until if meets a concelling function.

Another application consists of creating programmed cycles (eg. tapping cycles) with non-interruptible linked blocks. In the tapping application. The external FEED STOP can be taken to trigger a cycle withdrawal by using G10.

9.11 – READER OR DNC1 PASS MODE

This mode is used to machine while the program is either being read on the reader or transmitted by the DNC1. The mode is used with long programs which cannot be stored, or for machining a single part, for instance.

Program characteristics :

- Calls to sub–programs in memory (G77 H...) are allowed.

- Any reference to block numbers in the pass program **is not allowed** :

  . jumps to blocks G79 or G70N...
  . calls to a sequence of blocks G77N...N
  . declarations of emergency withdrawal blocks G75N...

NOTE :

The pass mode is indicated on the IN PROGRESS and LIST display pages, by PPR for reader pass mode and PPL for DNC1 pass mode.

9.12 – TEACH–IN FUNCTION

A program can be created or modified in teach–in mode. This function is carried out through "EDIT" mode, and the machine origins must have been set.

The "EDIT" mode provides access to the axis manipulators, enabling positioning of the axes.

The character "!" automatically recalls the curent point co–ordinates.

9.12.1 – Block modification

The co–ordinates contained in a block can be modified by the methods described in the "EDIT" mode. They can also be modified to integrate the current point values.

With the program selected and in "EDIT" mode, search for the block to be modified. Use the handlers to position the slide.

Enter	#! X Z LF	The X and Z co–ordinates are modified in the target block if they exist.
or		
	#! LF	All co–ordinates present in the block are modified (if X alone is programmed, it will be modified).
or		
	#! X LF...	X alone is modified, if present.

Where an axis has to be added to the block, the block must be brought into dialogue–line and the axis, preceded by "!", should be added

Example:	> N100 X20
	# LF
In dialogue–line:	# N 100 X20 ! Z LF (!Z added characters)

9.12.2 – Adding blocks

In "EDIT" mode, blocks may be added in an existing program.

Select the block behind which the new block is to be inserted, position the axis or axes at the point, and then, given that the current points of the axis names to the right of "!" are automatically entered, proceed as follows :

In dialogue–line : + N80 ! LF all co–ordinates are entered.

+ N90 F100 ! X LF

becomes

N90 F100 Xxxxx

+ N100 X150 ! Z LF

becomes

N100 X150 Zzzzz

9.12.3 – Creation of a program

In the "CHARGE" mode, key in a program. For example :

%2
M2
X OFF

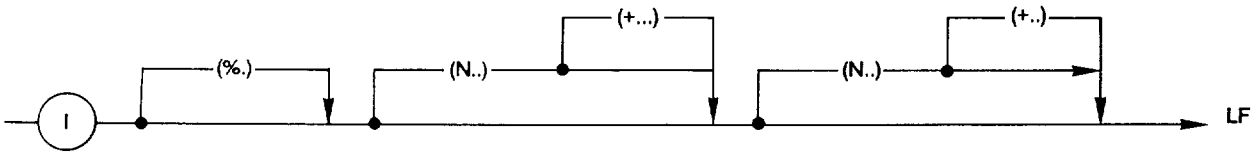
Then to "EDIT" mode, and continue as for "Adding blocks".

9.13 – INSERTION OF ALL OR PART OF A PROGRAM IN "EDIT" MODE

With this procedure you can duplicate a program, or insert parts of a program into another program.

Blocks are inserted in the "EDIT" mode after the block pointed by the cursor, using the character "I". The block number to be inserted must be less than 32767.

Access the "EDIT" mode, and select the program and sequence prior to insertion, then key in following the syntax below :



Any deviation from this syntax will cause the command to be denied, and the (dialogue) cursor will return to the beginning of the line, under the character "I".

When insertion is impossible due to shortage of storage capacity, the message "MODIFICATION IMPOSSIBLE" appears.

COMMENTS :

- When the program number %... is not specified, the program being edited is addressed.
- The pair of sequence numbers constitues the boundaries of the insertion zone. Both boundaries are included in the insertion.
- When a command defines a single sequence number N... the number addresses the start of the insertion zone, as the end boundary is also the end of the program itself.
- A boundary can be specified by a sequence number plus a value. For example, if N10 + 3 is programmed, the boundary is the 3rd block after the N10 sequence.

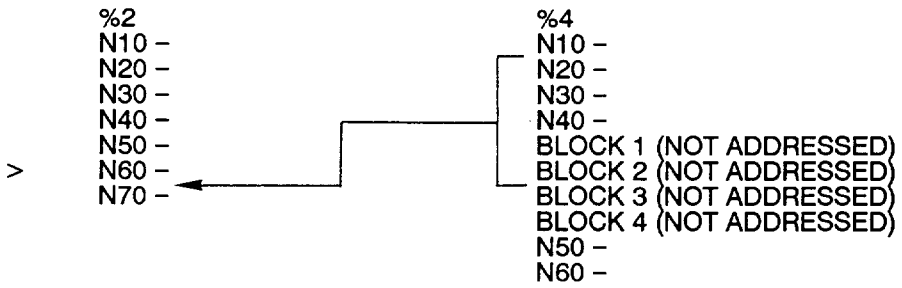
Examples :

- EDIT Mode

```
    %1
    N10 –
    N20 –
    N30 –
>   N40 –
    N50 –
```

Dialog line : I N10 N30 : Sequences 10 to 30 will be inserted after the sequence 40.

- EDIT Mode



Dialog line : I %4 N20 N40 + 2 : Sequences N20 to BLOCK 2 (NOT ADDRESSED) of program 4 will be inserted after sequence 60 of %2.

The block pointed to, "Np", must not be :

- inside the limits of the part of the program to be inserted,
- one of the limits of the part of the program to be inserted.

Example :

```
    %1
    N10
    :
    :
>   Np
    :
    :
    N50
    :
    :
    N100
```

I Np N50 is impossible. The message "MODIFICATION IMPOSSIBLE" is displayed.

9.14 - SETTING-UP OF "RECALL" MODE : FUNCTION M12

Function M12 is an M-function which is decoded "after". It sets up the "RECALL" mode, illuminating the FEED STOP and CYCLE indicators, and hands over to the axis handlers or manual manipulation crank, in unrestricted mode only, as the system remains in the program run "CONT SEQ" mode.

In TEST or RNS mode, this function is not available. When the operator unables FEED STOP this cancels function M12 and causes the program run to resume from the new slide position. There is NO axis recall.

Function M12 is present in the PLC on the W1.11 bit as long as the FEED STOP indicator is illuminated.

9.15 - POSITIONING AT A PROGRAMMED DISTANCE FROM A POINT

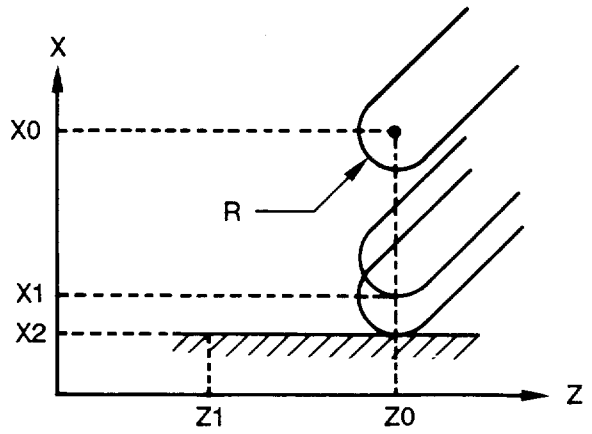
This function is active only with linear interpolation G0 or G1. The system positions the axis on the programmed path at a distance equal to the tool radius from point X, Z specified in the block, before the point if the sign following R is "-" or after if the sign is "+". This offset is applied only to the axes of the interpolation plane.

This block must not include any PGP, blends or chamfers.

Block syntax :

G1 (or G0) R+ (or -) X 053 Z 053

Example




```
N10 X0 Z0
N20 G0 R- X1
N30 G1 G42 X2 D...
N40 ...Z1
```

9.16 - TRANSFER OF THE CURRENT PARAMETER VALUES INTO THE PART PROGRAM : G76

Function G76 is used to update a file of E and L parameters with the new contents of the corresponding active data.

This file contains block sequences that can be stored in the part program itself or in a dedicated sub-program.

The block syntax is :

G76	Hp	Editing of the parameters of program %p.
	Nn1 Nn2	Editing of the parameters defined between Nn1 and Nn2 of the current program.
	Hp Nn1 Nn2	Editing of the parameters defined between Nn1 and Nn2 of program %p.

The parameters of a block are edited until a function other than declaration of an E or L parameter is encountered in the block.

EXAMPLE :

```
G76 N100 N120
N100 ..... (LF)
L101 = ..... E 80001      = ..... (LF)
L4   = ..... G4F6E52002 = ..... (LF)
L6   = ..... (LF)
G1 X100 L3 = ..... (LF)
N120 .....
```

Only parameters L101, E80001, L4 and L6 can be updated. The field designed to receive the parameter value (located after the symbol =) is an area of at least 10 characters including only spaces, decimal characters and possibly sign and decimal point.

Failure to comply with this format results in error 97.

PROGRAMMING EXAMPLE :

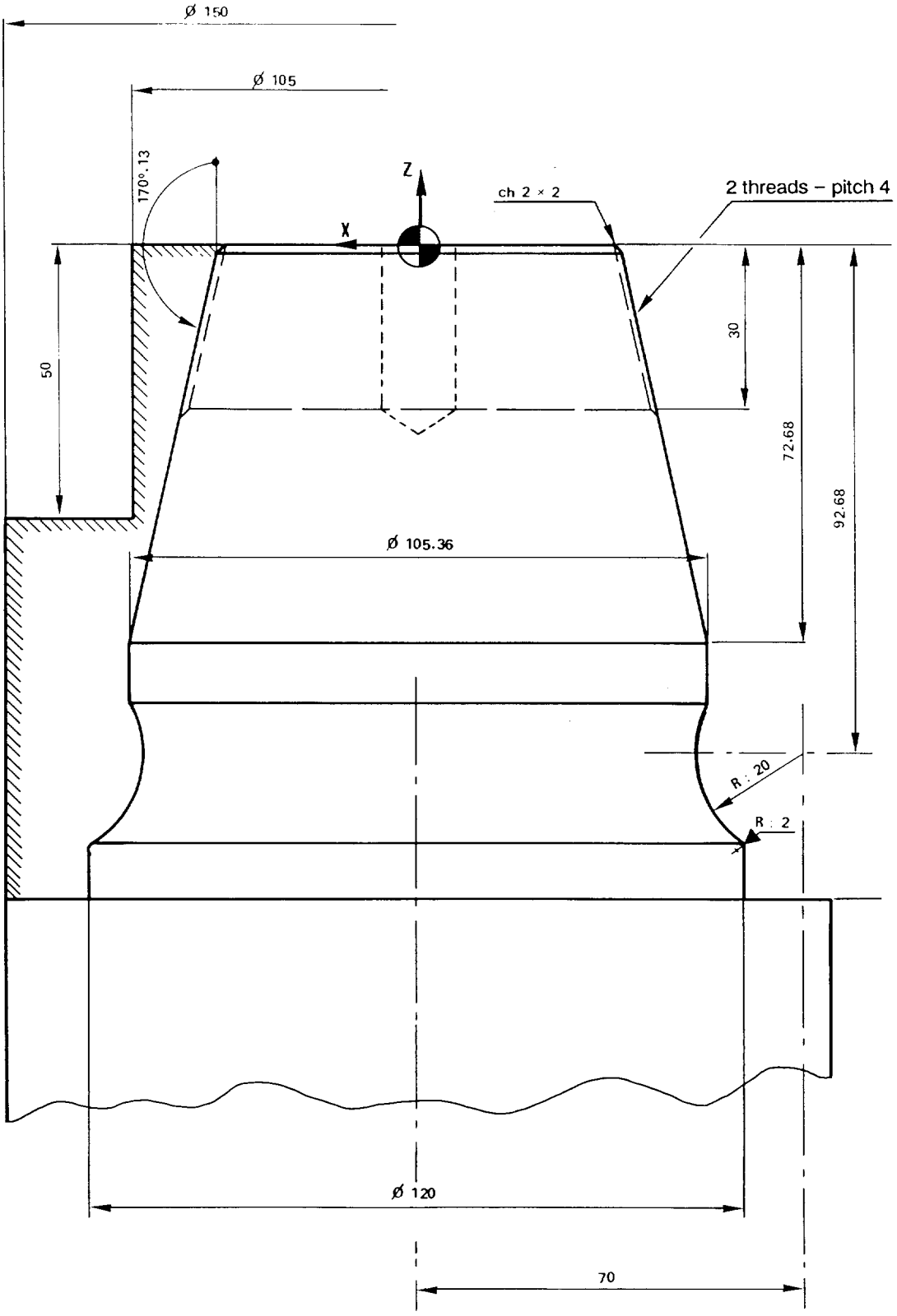
```
% 125
N10 G77 H200 N50 N80          Establish old data valves
    Execution of the program
    with modification of
    the parameters.
N600 G76 H200 N50 N80         Update of the data file
M02

% 200
N50
L1 = ~~~~~~~~~~~~~~~~~~~~  E52002 = ~~~~~~~~~~~~~~~~~~~~
E80004 = ~~~~~~~~~~~~~~~~~~~~  At least 10 characters or spaces
N80
```

When blocs N50 to N80 of program 200 are written, an initialisation value supplemented by spaces up to 10 characters or, if zero, by 10 spaces minimum is written after the symbol =.

At the beginning of program 125, the data from program 200 become active. At the end of program 125, program 200 is updated.

NOTES

10 – PROGRAMMING EXAMPLE



```
%21 (ROUGH TURNING CYCLE)
(ORIGIN OFFSETS AND TOOL LENGTHS)          Example of use of external parameters
E60000 = -214164    E61000 = -302865       DAT 1 input by the program
E50008 = 30778      E51008 = 43212         Tool dimension input on D8 (X and Z)
E50013 = 13335      E51013 = 30191         Tool dimension input on D13 (X and Z)
E50011 = 7656       E51011 = 182894        Tool dimension input on D11 (X and Z)

N10   G             X200    Z200
N15   G92           S2500
N20   T8            D8      M6
N30   M42           S1000   M3
N40   G79           N200

N50   G             X70     Z2
(FINISHED PROFILE DEFINITION)
N60   G1            Z0      F.15
N70   EA90          ES      EB-2
N80   X105.36       Z-72.68         EA170.13
N90   EA180ES
N100  G2            I140    K-92.68         R20     ES+ EB2
N110  G1            X120    Z-120   EA180
N120  X150

N200  G             X140    Z5
N202  G96           S600
(BLANK DEFINITION AND ROUGH TURNING EXECUTION)
N205  G64           N120    N50     I.5.K.1 P2
N210  G95           X150    Z-120   F.6
N213  Z-48
N216  X105
N220  Z2
N230  X70
N240  G80           X120

N250  G             X108    Z-81
(ROUGH TURN 20 RADIUS)
N255  G65           N110    N80     Z-110   EA-160  I.5     K.1     P2
N257  G             X110
N260  G             X200    Z200
N265  G42           X72     Z2
N267  G96           X72     S600
(FINISH PROFILE)
N270  G77           N60     N120
N275  G40           G       X200    Z200

N280  G97           S1300
N282  T11           D11     M6
N284  G             X       Z2
(DRILLING CYCLE)
N286  G83           Z-30    P10     Q5      F.1
N288  G80           G52     X-50    Z

N300  T13           D13     M6
N305  S1500
N310  X82.327       Z5
(THREADING CYCLE)
N320  G33  X90.507      Z-30    K4      EA-9.87  EB30    R3    P1.2    Q.06  S4    F2
N360  G             X200    Z200    M2
```

11 – STRUCTURED PROGRAMMING

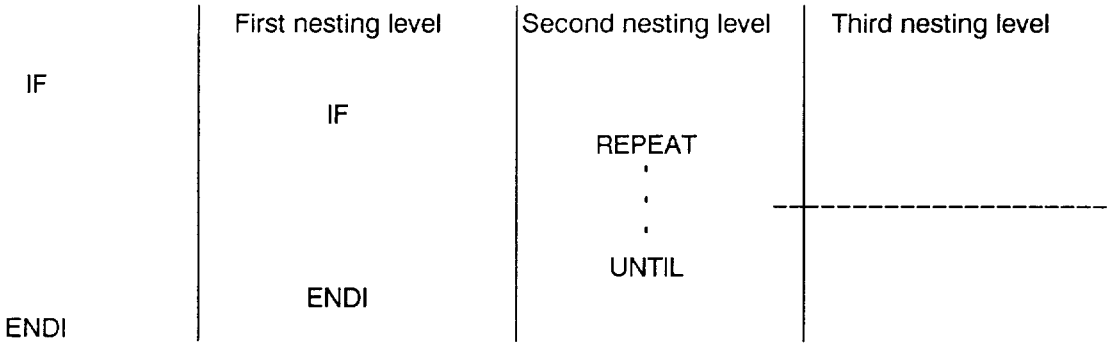
The system offers the possibility of programming structured jumps and loops, making the program easier to read and facilitating the programming of complicated part programs.

11.1 – STRUCTURE

A structured sequence always begins with one of the following keywords: IF, REPEAT, WHILE, FOR. It ends with ENDI for IF, UNTIL for REPEAT, ENDW for WHILE and ENDF for FOR. IF corresponds to a jump or branch, REPEAT, WHILE and FOR to a loop.

- An ELSE can be interposed between IF and ENDI.

- There are 15 possible nesting levels, independent of G77...



- Programming of a conditional or unconditional G79 is allowed in a structured sequence, but must mandatorily lead to the lowest nesting level of the current program or sub–program.

EXAMPLE :

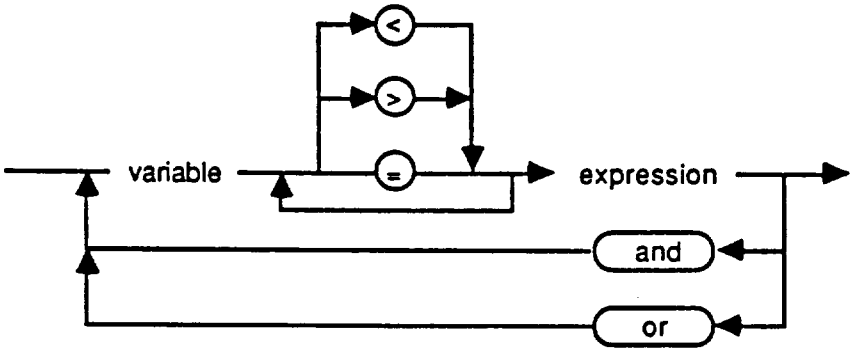
```
%1
IF
   WHILE
      G79 N100 allowed
      G77 H2
      G79 N50 error
      G79 N30 error
N30
   ENDW
N50
ENDI
N100
M2
```

```
%2
G79 N100 allowed
REPEAT
   IF
   G79 N100 allowed
   G79 N50 prohibited
   ENDI
N50
   UNTIL
N100
```

11.2 – LIST OF COMMANDS

11.2.1 – Definition of Terms

– Condition :



– Variable : All the L variables, symbolic variables and E parameters used for parametric programming.

– Expression : Sequence of parameters and immediate values linked by symbols +, –, *, /, !, &.

The calculations are made sequentially.

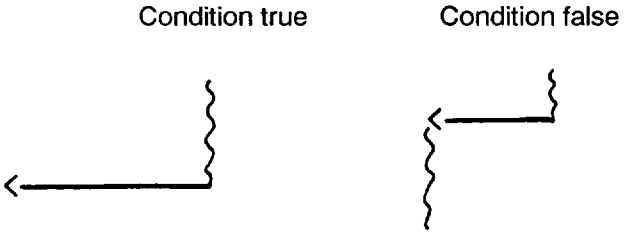
11.2.2 – IF

IF condition THEN (1)

(ELSE)

ENDI

ELSE is optional.

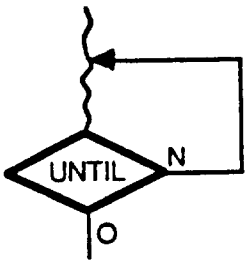


(1) THEN can also be programmed at the beginning of the next block and followed by functions to be executed in that block :

For instance : IF L4 < 8
THEN L6 = L2 + 1 X L6

11.2.3 – REPEAT : Equivalent to a Loop

REPEAT
UNTIL condition

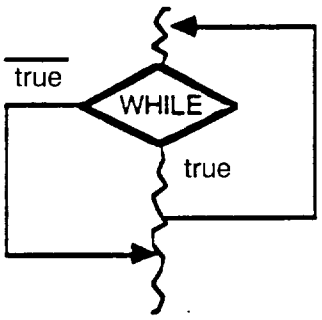


11.2.4 - WHILE

WHILE condition DO
ENDW

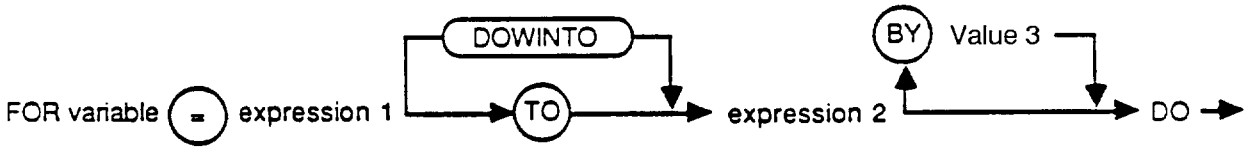
DO can also be programmed at the beginning of the next block and followed by functions to be executed in that block :

For instance : WHILE condition
DO X L6



If DO is omitted after WHILE, the system generates an error message (E191).

11.2.5 - FOR



- The variable is an L function, a symbolic variable or a parameter E80000, E81000 or E82000 (be careful with stopping of calculations for L100 to L199 and E...).

  The values of the expressions are positive or negative integers. The system will round them off (to the next lower digit up to 0.5 and the next higher digit above 0.5).

- With the TO function, the loop is not executed when the current value of the variable becomes higher than the final value (incrementing of the variable).

- With the DOWNTO function, the loop is not executed when the current value falls below the final value (decrementing of the variable).

- The default value of BY is +1 to increment or decrement the variable each iteration. The value of BY must always be positive. If it is negative, functions TO and DOWNTO are reversed.

11.2.6 - EXIT

The EXIT instruction is used to exit from the current iteration loop and go up to the next nesting level (FOR, WHILE or REPEAT loop), bypassing the IFs inside this loop.

It must be included in an IF --THEN or ELSE-- condition.

EXAMPLES :

```
REPEAT --                                  WHILE
  ------                                     REPEAT
  ------                                       IF
IF                THEN condition               IF
  ------                                         IF
  ------                                           EXIT ----------
ELSE                                               ENDI          |
EXIT                                             ENDI            |
ENDI                                           ENDI              |
  ------                                     ENDI                |
  ------                                   UNTIL <---------------
UNTIL             condizione             ENDW
```

11.3 - SYNTAX

- The words IF, REPEAT, WHILE, FOR, ELSE, ENDI, UNTIL, EXIT, ENDW and ENDF must be the first in a block (they must not be preceded by sequence numbers).
- The condition must follow words IF, WHILE and UNTIL in the same block. Similarly, the limits and possibly the increment must follow the word FOR.
- The words DO and THEN must immediately follow the condition. If they are not in the same block as IF, WHILE or FOR, they must be the first words in the next block.
- The words DO, THEN and ELSE can be followed by ISO functions in the same block.

  Example : WHILE L1 < 3 DO G91 X 12
  or
  WHILE L1 < 3
  DO G91 X 12

The following programming is refused : WHILE L1 < 3 G91 X 12
DO
prohibited in the condition

- Blocks beginning with END, ENDW, ENDF, EXIT or UNTIL must not include any ISO functions.
- The space is mandatory after IF, REPEAT, THEN, ELSE, UNTIL, WHILE, DO, and DOWNTO.

  For example : WHILELO < 3 is not recognised by the system.
- ISO blocks with sequence numbers are allowed in the loops.

12 - PROGRAM STACK - SYMBOLIC VARIABLES

Symbolic names allow symbolic designation of the variables used in parametric programming and extension of the number of variables.

Each axis group (including graphic axes) is assigned a stack at the bottom of the memory to save the L variables and allocate symbolic variables.

12.1 - STACK ALLOCATION

The stack is allocated in MDI by declaring the size in bytes in square brackets after %.

Example : % [2000] (LF) (cycle)

Allocation of 2000 bytes in each stack : one stack for the axis group and one for the graphic axis.

The stack size is fondamental. The only way to change it is by making a new allocation. The contents of the stack are destroyed by a reset but not the space allocated.

12.2 - SAVE AND RESTORE OF L VARIABLES

PUSH Lm - Ln

Save on the stack the variables Lm to Ln (including limits), where m and n are numbers within one of the following ranges: 0-19 or 100-199 or 900-939.

PUSH L1-L110 is prohibited. Instead, program

```
PUSH L1-L19
PUSH L100-L110
```

PULL Lm - Ln

Recovery of Lm to Ln of the last values saved for the same variable set Lm to Ln.

```
%1                                          %1
'                                           '
'                                           '
L0 = 1                                      L0 = 1
'                                           '
'                                           '
PUSH L0-L10                                 PUSH L0-L10
'                                           '
'                                           '
L0 = 2                                      L0 = 2
'                                           '
'                                           '
PUSH L0-L10                                 PUSH L0-L11
'                                           '
'                                           '
PULL L0-L10   }  Recovery of                PULL L0-L11 => L0 = 2
'             }  the last PUSH              '
'             }  L0 = 2 in                  '
PULL L0-L10   }  both cases                 PULL L0-L10 => L0 = 1
```

A PULL restores only the variable set saved in the current program or sub-program or from previous nesting levels.

```
%1                                          %2
'                                           '
'                                           '
PUSH L0-L10                                 PULL L0-L10
'                                           '
'                                           '
G77  H2                                     G77  N1  N2
'                                           '
'                                           '
                                            PULL L10-L12
                                            '
                                            '
                                            N1
                                            '
                                            '
                                            PUSH L10-L12
                                            '
                                            '
                                            N2
```

prohibited

The words PUSH and PULL must be the first words in a block and must be followed by a space (the block must not begin with the sequence number).

12.3 - SYMBOLIC VARIABLES

12.3.1 - Declaration of the Variables

The variables are declared between keywords VAR and ENDV.
VAR and ENDV must be first words in a block. VAR must be followed by a space.

Preparation and execution of the blocks are synchronised, except in M999.
The symbols are declared between square brackets : [...].

A symbol includes from 1 to 8 alphanumeric characters. The first character must always be alphabetic.

Example : VAR [INDICE] [PIPI]
[FASE]
ENDV (only word in the block)

To define an array of variables, the array dimensions are included in round brackets after the last character.

An array can include up to four dimensions. The symbol "," or ";" separates the values of the different dimensions.

Example : [TAB1 (10)] [TAB2 (2, 5, 3)]

Array 2 is allocated 2 x 5 x 3 = 30 entries, i.e. 5 groups of 2 arranged in 3 groups of 10.

The value of each dimension must be between 1 and 65535 for an array with 1 dimension and between 1 and 255 for array with 2, 3 or 4 dimensions.

The value of a dimension must be declared as an immediate value or a symbolic variable that has already been initialised. (Dimensions cannot be defined by L variables or E parameters.)

Example : [VAR1] = 10]
[VAR] = [TAB1 (VAR1,5)] is equivalent to [TAB1 (10,5)]

Symbolic variables have real values.

12.3.2 - Static Initialisation of the Variables and Arrays

The default value of the variables and arrays is zero.

They can be initialised by declaring the character = followed by the initial value(s) separated by "," after the symbol.

The initial values can be declared over several blocks. In this case, the character = must be repeated before the value in the new block.

Example : VAR [NBT] = 4 [TABLE (2, NBT (2, NBT)] = 3,6 (LF)
= 10, 1, 8, 2, 6, 6 (LF)
ENDV

If L0 = [TABLE (2,3)]
L0 = 2 (sixth digit)

12.3.3 – Use of Symbolic Variables in an ISO Program

Symbolic variables can be assigned to all the ISO functions and can be used in parametric expressions associated or not with L variables and E parameters. They can also appear in structured branch and loop commands.

The array indexes are immediate values or symbolic variables (parameters L and E are prohibited). If [TAB (L0, 3)] is programmed, it is variable [L0] and not parameter L0 that is sought.

Array indexes can also be the additions and/or subtractions of values or symbolic variables.

```
Example : VAR[IX]  [COSX]  [SINX]  [NBT] = 4
              [TAB (2, NBT)] = 0, 0, 10, 5, 20, 8, 30, -2
          ENDV
          FOR[IX]= 1 TO [NBT] - 1 D0
              [COSX] = [TABL (1,IX+1)] - [TAB (1, IX)]
              [SINX] = [TABL (2,IX+1)] - [TAB (2, IX)]
                L0 = [COSX]*[COSX] LO = [SINX]*[SINX]+ L0
              [COSX] = [COSX]/ RLO   [SINX]=[SINX]/ RLO

               X[TAB(1,IX+1)]       Y[TAB(2,IX+1)]
          ENDF
```

- Only the variables declared in the current program or sub-program and those of previous nesting levels can be accessed.

12.3.4 – STRUCTURE OF MULTIDIMENSIONAL ARRAYS

The entries of the first dimension are located at the beginning of the array and are multiplied by the entries of the second dimension, the result being multiplied by the number of entries of the following dimension, etc.

12.4 – CREATION OF AN ARRAY FOR STORAGE OF A PROFILE – BUILD FUNCTION

The system offers the possibility of storing a profile written in ISO or PGP in an array on the stack. The array is created as the blocks of the profile are read. The executable blocks are stored in a two-dimensional array, whose first dimension is the functions to be saved in a block and whose second dimension is the number of blocks of the profile.

The data of this array can then be accessed by parametric programming.

12.4.1 – Data of a Block that Can Be Stored on the Stack

- G functions :

  Only one G function per block can be stored : it is the modal function G0, G1, G2 or G3 that is stored in the array.

- Axis dimensions :

  Depending on machine parameter P0 :

  . functions X, Z, U, W, C

    The modal values of these functions are stored in the array. The format for the dimensions is 5.3 (inches or tenths of micron not taken into account).

- Other dimensions :

  . I, K, P, Q, R

    Their values are not stored in the array unless they are programmed in the block. If they are not programmed, the value 0 is stored in these entries. The format of these dimensions is 5.3.

- Feed rate – Function F :

  The modal value of function F is stored in the array. The format is 5.2 in mm/min or 2.3 in feed per revolution.

- T function :

  The T function is not stored in the array unless it is programmed in the block. Otherwise, the value 0 is initialised.

  The T function has an absolute value of 5 decimal digits maximum.

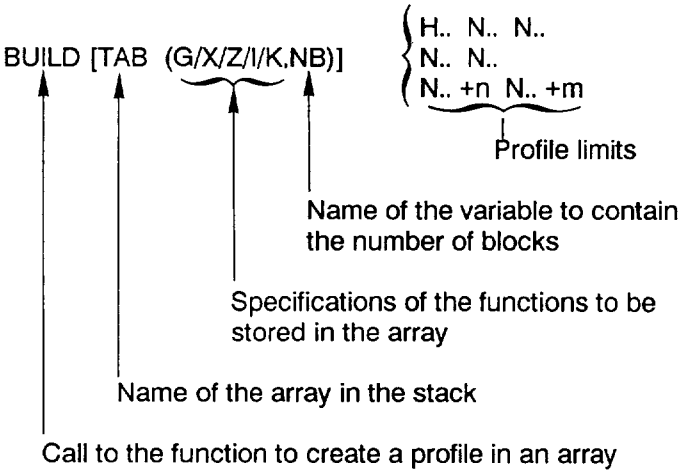
- S function :

  The S function is not stored in the array unless it is programmed in the block. Otherwise, the value 0 is initialised. The S function has an absolute value of four decades maximum or the format 2.2, depending on the spindle characteristics (see machine parameter P7).

REMARK :

Error 1 is reported if the format of the variable is incorrect ; e.g. F [FEED RATE] with [FEED RATE] = 123456.

12.4.2 – Programming



The two-dimensional array TAB and variable NB are automatically created by this function.

- The word BUILD followed by the array declaration must be programmed at the beginning of a block and be separated by at least one space.

- In the first dimension of the array, additional entries, not initialised by the data in the blocks, can be allocated; They are declared by "0"s separated by "/".

Example : [PROF1 (G/X/Y/Z/O/O,NBP)]
the first dimension of PROF1 has 6 entries.

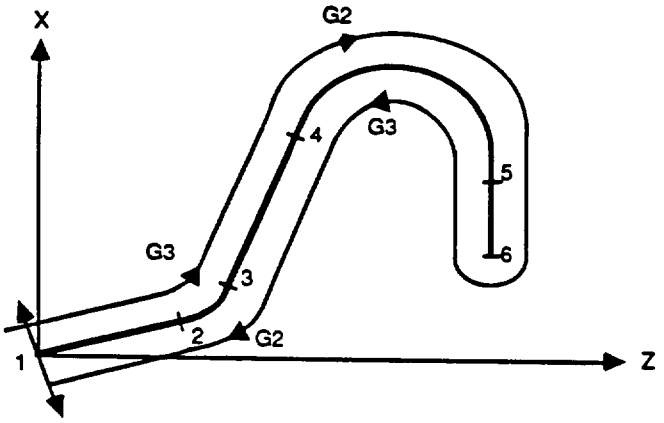
- The maximum number of entries that can be declared in the first dimension is 16.

- The maximum number of blocks that can be stored in the array is 255 (limit when there are several dimensions).

12.4.3 – Programming Errors

E.195 : Program stack saturated
E.196 : Noncompliance with array dimension limits
E.198 : Error in declaration of the array or variable symbol (the first character of a symbol must always be alphabetic)
E.199 : Syntax error in variable and array declaration (square brackets, round brackets, commas)
E.82 : First or last block of the profile incomplete (PGP)

12.4.4 – Example : Curve drawn from point 1 to point 6 and return to 1 with radius compensation



Pts

	G	X	Z	I	K
1	1	0	0	0	0
2	1	8.104	22.266		
3	3	14.081	28.243	17.501	18.846
4	1	35.13	35.905	0	0
5	2	30	65	30	50
6	1	20	65	0	0

[NB]-1

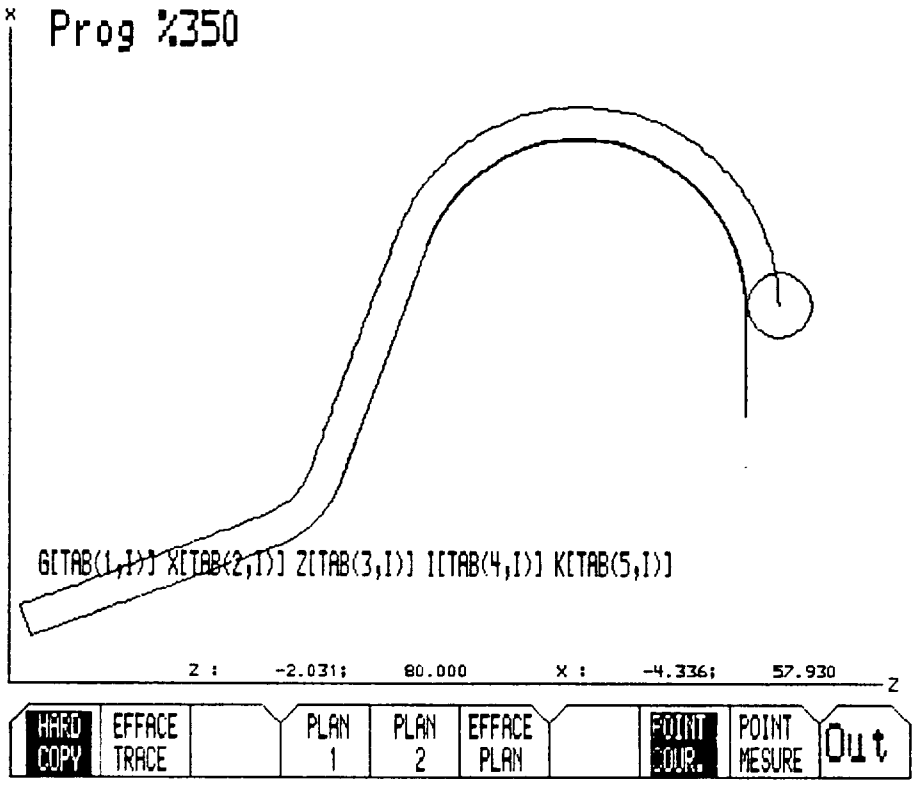
G [TAB (1,I)]

X [TAB (2,I)]

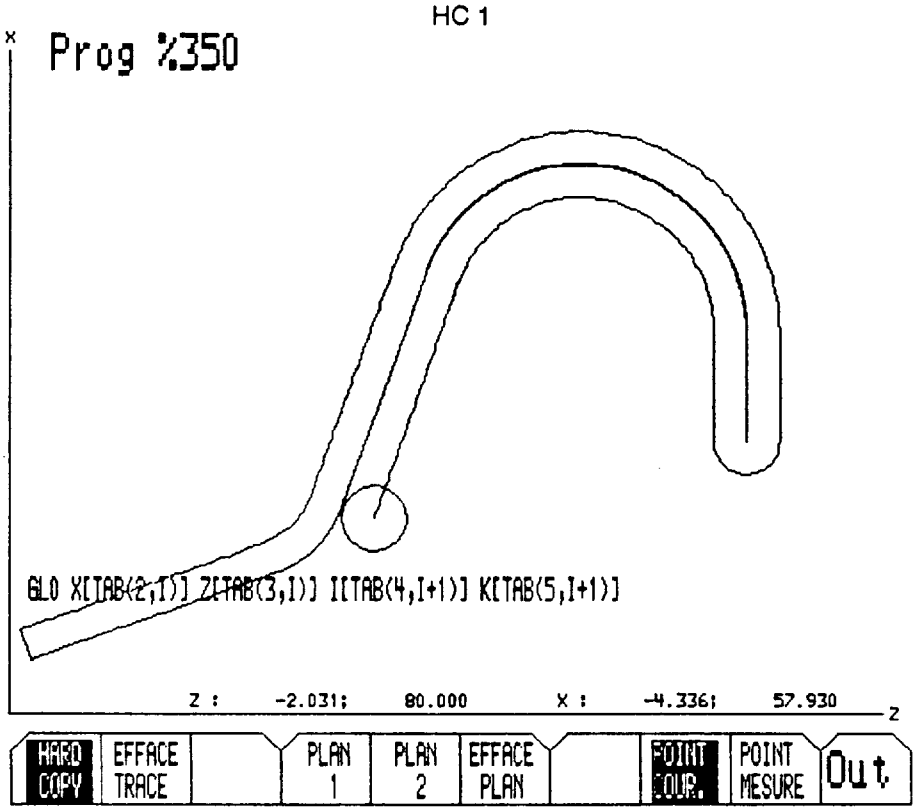
Value stored in the array when BUILD is executed

```
% 350
Z X
M999 G79 N100
N10   G1 X Z
      EA20 ES EB10
      EA70
      G2 I30 K50 R15
N40 G1 X20
N100
BUILD [TAB (G/X/Z/I/K,NB)] N10 N40

VAR [I]
ENDV
N115
FOR [I] = 1 TO [NB] DO G41 D1  (Movement along the curve from 1 to 6)
G[TAB (1,I)] X [TAB(2,I)] Z[TAB(3,I)] I[TAB(4,I)] K[TAB(5,I)]
ENDF
FOR [I] = [NB] -1 DOWNTO 1 DO (Movement along the curve from 6 to 1)
LO = [TAB(1, I+1)]
IF LO>1 THEN LO = LO*3+1&3      Reversal of G2 and G3 for the path from 6 to 1
ENDI
GLO X [TAB(2,I)] Z [TAB(3,I)] I[TAB(4,I+1)] K[TAB(5,I+1)]
ENDF
G40 X Z
M2
```

Definition of the curve (points 1 to 6)


Prog %350
G[TAB(1,I)] X[TAB(2,I)] Z[TAB(3,I)] I[TAB(4,I)] K[TAB(5,I)]
Z : -2.031; 80.000 X : -4.336; 57.930
HARD COPY
EFFACE TRACE
PLAN 1
PLAN 2
EFFACE PLAN
POINT COUR.
POINT MESURE
Out

HC 1


Prog %350
GLO X[TAB(2,I)] Z[TAB(3,I)] I[TAB(4,I+1)] K[TAB(5,I+1)]
Z : -2.031; 80.000 X : -4.336; 57.930
HARD COPY
EFFACE TRACE
PLAN 1
PLAN 2
EFFACE PLAN
POINT COUR.
POINT MESURE
Out

HC 2

12.5 - DELETION OF VARIABLES AND ARRAYS FROM THE STACK

The deletion is not executed until after execution of the block preceding the deletion command.

12.5.1 - Automatic Variable Deletion

- The return from a subroutine deletes all the variables declared in this subroutine and frees the space occupied on the stack.

- The end of the program (M02) or an operator reset deletes all the variables and resets the stack.

12.5.2 - Programmed Deletion of Variables

Variable deletion is programmed by the DELETE function followed by the list of variables and arrays between square brackets, separated by "/".

Example : DELETE [IX]/[TAB1]/[PROF1]

- The list of variables to be deleted must be separated from the word DELETE by at least one space. However, there must be no spaces in the list of variables.

- DELETE cannot delete the variables of the calling program.

- The word DELETE must be the first in the block.

REMARK :

In recognition of keywords by the system, only the first four characters are significant.

For instance, the command :

DELE [IX]/[TAB1]/[PROF1] is accepted.

NOTES

APPENDIX 1

LIST OF ERRORS

This full list of errors may not apply to all installations

* E.1 : UNACCEPTABLE CHARACTER
* AXIS NOT RECOGNIZED BY THE SYSTEM?
* TOO MANY DIGITS AFTER A FUNCTION?
* A SIGN WITH A FUNCTION WHICH SHOULDN'T HAVE ONE?
* INCORRECT FORMAT?
SYMBOLIC VARIABLE FORMAT ERROR E.G.: F[FEED RATE] WHERE [FEED RATE] = 123456

* E.2 : G FUNCTION NOT RECOGNIZED BY THE SYSTEM
*
*
*

* E.3 :
*
*
*

* E.4 : PARAMETERIZED PROGRAMMING OPTION NOT ENABLED
* LETTER O PROGRAMMED IN PLACE OF ZERO
*
*
*

* E.5 : GEOMETRIC PROGRAMMING OPTION NOT ENABLED
*
*
*

* E.6 :
*
*
*

* E.7 :
*
*
*

* E.8 : TOOL COMPENSATION NO. TOO BIG
*
*
*

* E.9 : TOO MANY CONSECUTIVE NON WORKING BLOCKS
* ENDLESS CALCULATION LOOP?
*
*
*

```
* E.10      :
*
*
*

* E.11      :
*
*
*

* E.12      :
*
*
*

* E.13      :
*
*
*

* E.14      :
*
*
*

* E.15      :
*
*
*

* E.16      :
*
*
*

* E.17      :  MISSING CLOSE BRACKET
*
*

* E.18      :
*
*
*

* E.19      :
*
*
*
```

* E.20 : M02 MISSING AT END OF PROGRAM
*
*
*

* E.21 :
*
*

* E.22 :
*
*
*

* E.23 :
*
*
*

* E.24 :
*
*
*

* E.25 : UNDEFINED SUB-PROGRAM OR SEQUENCE NUMBER
*
*
*

* E.26 : TOO MANY SUB-PROGRAM NESTINGS LEVELS
*
*
*

* E.27 : RADIUS COMPENSATION
* - IN MACHINE ORIGIN PROGRAMMING (G52),
* - OR DURING AN INTERRUPTABLE MOVE (G10),
* - OR WITH M00 OR M01 IN THE BLOCK,
* - OR IN A DRILLING CYCLE

* E.28 : SYNTAX ERROR IN CONSTANT CUTTING SPEED MODE OR IN DEFINING
* THE RADIUS
* - G96 MUST BE FOLLOWED BY "S"
* - G97 MUST BE FOLLOWED BY "S"
* - "X" MUST BE PROGRAMMED BEFORE G96 IN THE BLOCK
* - THE ONLY AXIS ALLOWED WITH G96 IS "X".
* - "X" NOT REPROGRAMMED UPTO G96, AFTER G52

* E.29 : NO RANGE PROGRAMMED IN CONSTANT CUTTING SPEED MODE,
* OR, "S" NOT IN THIS RANGE, IN G97
* - WITHOUT RANGE-SEARCH OPTION : S NOT INCLUDED BETWEEN THE MINIMUM AND
* MAXIMUM OF THE RANGE PROGRAMMED.
* - WITH RANGE-SEARCH OPTION : S DOES NOT BELONG TO ANY RANGE.

*--- OPERATIONAL FAULTS (E30 to E33 – E36 – E40 to E42)

* E.30 : READER UNCONNECTED OR FAULTY
* OR DNC LINE FAULT
* OR PLOTTER FAULT
*
*
* E.31 : DATA EXCHANGE WITH OPERATOR PANEL FAULT
*
*
*

* E.32 : AXIS REFERENCING ERROR (SLIDE ALREADY ON ITS SWITCH)
*
*
*

* E.33 : FAULT IN FIBRE-OPTIC LINK
*
*

* E.34 :
*
*

* E.35 : SEARCHED FOR SEQUENCE NO. NOT FOUND
*
*
*

* E.36 : WORKPIECE PROGRAM MEMORY FULL
* MEMORY AREA INSUFFICIENT FOR PASS MODE
*
*

* E.37 : MAX. THREADING VELOCITY EXCEEDED
*
*
*

* E.38 :
*
*
*

* E.39 :
*
*
*

* E.40 : EXCESSIVE SERVO FOLLOWING ERROR – AXIS 0
*
*
*

* E.41 : EXCESSIVE SERVO FOLLOWING ERROR – AXIS 1
*
*
*

* E.42 : EXCESSIVE SERVO FOLLOWING ERROR – AXIS 2
*
*
*

* E.43 :
*
*
*

* E.44 :
*
*
*

* E.45 :
*
*
*

* E.46 :
*
*
*

* E.47 :
*
*
*

* E.48 :
*
*
*

* E.49 :
*
*
*

```
*---        LIST OF THREADING ERRORS

* E.50    :  "K" MISSING FROM 1ST BLOCK IN G38 ?
*            AT LEAST ONE OF "X, Z, K, OR P", DIMENSIONS MISSING IN G33 ?
*
*

* E.51    :  THE PRESENCE OF AT LEAST ONE ADDRESS, OTHER THAN
*            X,Z,K,P,Q,EA,EB,R,F,S, IN G33
*
*

* E.52    :  ANGLES EA AND EB ARE INCOMPATIBLE, IN G33
*
*
*

* E.53    :
*
*
*

* E.54    :  EB >= 90   OR   EB <= -90 IN G33
*
*
*

* E.55    :  P <= 0,  R < 0,  Q < 0,  F <= 0,  S <= 0,  K <= 0, IN G33
*
*
*

* E.56    :  ZERO CLEARANCE OR LENGTH OF PULLOUT >= THREAD LENGTH, IN
*            G33
*
*

* E.57    :  Q >= P, IN G33
*
*
*

* E.58    :  PRECEDING BLOCK, INCOMPATIBLE WITH G33
*
*
*

* E.59    :  F >= 100,  S >= 100,  K > 250,  IN G33
*
*
*
```

* E.70 : X & Z AXES NOT YET PROGRAMMED WHEN G33 CALLED
*
*
*

* E.71 : INCOMPLETE DATA, RELATIVE TO THE PREVIOUS BLOCK
*
*
*

* E.72 :
*
*
*

* E.73 :
*
*
*

* E.74 :
*
*
*

* E.75 :
*
*
*

* E.76 :
*
*
*

* E.77 :
*
*
*

* E.78 :
*
*
*

* E.79 :
*
*
*

* --- PROGRAMMING ERRORS IN ROUGH TURNING CYCLES

* E.80 : – AN INCOMPLETE BLOCK PRECEEDS THE CYCLE CALL
* – PROGRAMMED CIRCLE OR INCOMPLETE BLOCK IN THE BLANK DEFINITION
* – GROOVE CYCLE IMPOSSIBLE (BAD FORMAT OR NO POINT OF INTERSECTION)

* E.81 : ROUGHING DEPTH OF CUT, NOT DEFINED
*
*
*

* E.82 : THE 1ST OR LAST BLOCK OF THE FINISH PROFILE HAS NO DIMENSIONS OR IS INCOMPLETE
*
*

* E.83 : THE FINISH PROFILE HAS TOO MANY BLOCKS (MAX. 50)
*
*
*

* E.84 : GROOVE CYCLE : INTERSECTION POINT NOT FOUND
*
*
*

* E.85 :
*
*
*

* E.86 : SYNTAX ERROR IN DEFINITION OF PLUNGING CYCLE
* EF PROGRAMMED BEFORE EB (+ OR –)
*
*

* E.87 : SYNTAX ERROR IN THE DEFINITION OF A CYCLE :
* – M3 OR M4 MISSING?
* – P AND/OR Q, ZERO OR MISSING IN G83 OR G87?
* – P INCOMPATIBLE WITH THE CYCLE?
* – DWELL INCOMPATIBLE WITH THE CYCLE?

* E.88 :
*
*
*

* E.89 :
*
*
*

* E.91 : UNRECOGNISED PARAMETER/VARIABLE
*
*
*

* E.92 : ERROR IN ASSIGNING A PARAMETER/VARIABLE
* – AN UNSIGNED FUNCTION ASSIGNED A NEGATIVE PARAMETER?
* – THE PARAMETER VALUE EXCEEDS THE ASSIGNED FUNCTION LIMIT?
*

* E.93 : ERROR IN THE DEFINITION OF A PARAMETER OR IN THE
* EXPRESSION OF A TEST :
* – RELATIONAL SYMBOL "=,<,>" FOLLOWING AN L VARIABLE
* MISSING?
* – ONE OF THE CHARACTERS "–,+,*,/" UNRELATED TO ITS
* FUNCTION?

* E.94 : IMPOSSIBLE OPERATION IN AN EXPRESSION :
* – SQUARE ROOT OF A NEGATIVE NUMBER?
* – DIVISION BY 0?
*

* E.95 : ATTEMPT TO WRITE TO A READ ONLY PARAMETER
*
*

* E.96 : AN EXTERNAL PARAMETER DEFINITION (WHICH SUSPENDS BLOCK
* PROCESSING)
* PROGRAMMING OF L100 TO L199 OR L900 TO L939
* IN G64 PROFILE DEFINITION

* E.97 : UPDATE OF THE PARAMETER IMPOSSIBLE IN G76 :
* – NO "=" SYMBOL AFTER THE PARAMETER NUMBER
* – LESS THAN 10 CHARACTERS ALLOCATED FOR THE VALUE
*

* E.98 :
*
*
*

* E.99 :
*
*
*

* --- ERRORS GENERATED FROM GPP PROFILE DEFINITIONS

* --- FROM BLOCKS IN WHICH THE END POINT SHOULD BE DIRECTLY DETERMINED

* E101 : INSUFFICIENT DATA IN THE PROGRAMMING OF A CIRCLE
*
*
*

* E102 : WHEN PROGRAMMING A LINE USING ITS ANGLE (EA) AND ONE
* COORDINATE, (X, Y OR Z,) THE MISSING COORDINATE IS
* IMPOSSIBLE TO CALCULATE.
*

* E103 :
*
*
*

* E104 :
*
*
*

* E105 :
*
*
*

* E106 :
*
*
*

* E107 : – PROGRAMMING A CIRCLE USING ITS RADIUS AND END POINT, IN
* WHICH THE DISTANCE BETWEEN THE START AND END POINTS
* IS MORE THAN TWICE THE RADIUS.
* – OR, PROGRAMMING A CIRCLE USING X, Z, I, K, IN WHICH THE
* TRAJECTORY MISSES THE END POINT BY MORE THAN 20 MICRONS.

* E108 :
*
*
*

* E109 :
*
*
*

```
* ---      ERRORS FROM PROFILES WHOSE END POINTS SHOULD BE RESOLVED OVER TWO
           BLOCKS

* E110   :  SYNTAX ERROR IN THE FIRST OF THE TWO BLOCKS
*
*
*

* E111   :  SYNTAX ERROR IN THE SECOND BLOCK
*
*
*

* E112   :  LINE-LINE INTERSECTION IN WHICH :
*             - START POINT OF 1ST BLOCK IS COINCIDENT WITH END
*               POINT OF 2ND BLOCK
*             - OR ANGLE OF 1ST LINE = ANGLE OF 2ND LINE

* E113   :  THE INTERSECTION OR TANGENCY POINT CANNOT BE CALCULATED
*           USING THE VALUES PROGRAMMED IN THE TWO BLOCKS
*
*

* E114   :  THE POINT OF INTERSECTION OR TANGENCY IS NOT
*           DEFINED BY ET+, ET-, ES+ OR ES-
*
*

* E115   :
*
*
*

* E116   :
*
*
*

* E117   :
*
*
*

* E118   :
*
*
*

* E119   :
*
*
*
```

* --- ERRORS FROM PROFILES WHOSE END POINTS SHOULD BE RESOLVED OVER THREE BLOCKS

* E121 : SYNTAX ERROR IN LAST 0F 3 BLOCKS
*
*
*

* E122 : 1ST TWO BLOCKS ARE NON-SECANT STRAIGHT LINES
*
*
*

* E123 : THE POINTS OF TANGENCY CANNOT BE CALCULATED FROM THE DATA
* IN THE 3 BLOCKS
*
*

* E124 : THE POINT OF TANGENCY BETWEEN THE 2ND AND 3RD BLOCKS
* IS NOT DEFINED BY ET+ OR ET-
*
*

* E125 :
*
*
*

* E126 :
*
*
*

* E127 :
*
*
*

* E128 :
*
*
*

* E129 :
*
*
*

* --- ERRORS IN THE DEFINITION OF A BLEND RADIUS OR CHAMFER

* E130 : NO AXIS MOVEMENT IN THE FIRST OF THE TWO BLOCKS
* TO BE CONNECTED BY A FILLET RADIUS OR A CHAMFER
*
*

* E131 : A FILLET OR CHAMFER MUST NOT BE PROGRAMMED
* IN A BLOCK WITH M00 OR M01 OR M02.
* – OR, THE END POINT CANNOT BE CALCULATED FROM
* THE GIVEN DATA BLOCKS

* E132 :
*
*
*

* E133 :
*
*
*

* E134 :
*
*
*

* E135 : A CHAMFER CAN ONLY CONNECT TWO STRAIGHT LINES
*
*
*

* E136 : MORE THAN TWO NON–MOVEMENT BLOCKS BETWEEN TWO GEOMETRIC
* ELEMENTS WHOSE POINT OF INTERSECTION OR TANGENCY IS TO BE
* CALCULATED
*

* E137 :
*
*
*

* E138 :
*
*
*

* E139 :
*
*
*

* E140 : ERROR IN RADIUS COMPENSATION MODE :
* – TOO MANY BLOCKS BETWEEN TWO COMPENSATED MOVES?
* – THE FOLLOWING FUNCTIONS ARE NOT ALLOWED IN G41/2 :
* M00, M01, M02; ACCESS TO EXTERNAL PARAMETERS; WRITING
* TO E8xxxx OR L1xx PARAMETERS

* E141 :
*
*
*

* E142 :
*
*
*

* E143 :
*
*
*

* E144 :
*
*
*

* E145 :
*
*
*

* E146 :
*
*
*

* E147 :
*
*
*

* E148 :
*
*
*

* E149 : TOOL CORRECTION EXCESSIVE WITH RESPECT TO THE PATH REQUESTED
*
*
*

```
* ---          MOVE DEMANDED BEYOND A SLIDE CAPACITY

* E150     :   DEMAND EXCEEDS TRAVEL LIMIT ON AXIS 0
*
*
*

* E151     :   DEMAND EXCEEDS TRAVEL LIMIT ON AXIS 1
*
*
*

* E152     :   DEMAND EXCEEDS TRAVEL LIMIT ON AXIS 2
*
*
*

* E153     :
*
*
*

* E154     :
*
*
*

* E155     :
*
*
*

* E156     :
*
*
*

* E157     :
*
*
*

* E158     :
*
*
*

* E159     :   MOVE DEMANDED ON A NON-REFERENCED AXIS
*
*
```

```
* ---        STRUCTURED PROGRAMMING ERRORS

* E190    :  TOO MANY BRANCH OR LOOP NESTING LEVELS (MAX. 15)
*
*
*

* E191    :  STRUCTURED PROGRAMMING SYNTAX ERROR
*
*           STRUCTURED PROGRAMMING PROHIBITED IN MDI
*
*           THE LOOP INDEX MUST BE AN L VARIABLE,
*           A SYMBOLIC VARIABLE OR A PARAMETER E80000 OR E81000 OR E82000
*           DO OMITTED AFTER WHILE
*           SYNTAX ERROR IN PUSH AND PULL

* E192    :  KEYWORD NOT RECOGNISED OR PROHIBITED IN THE PROGRAM CONTEXT
*
*
*

* E193    :  STRUCTURE ERROR
*
*
*

* E195    :  PROGRAM STACK SATURATED
*
*           MORE CONSTANTS DEFINED THAN ALLOCATED
*

* E196    :  ARRAY INDEX DECLARATION ERROR
*
*
*

* E197    :  USE OF A SYMBOL NOT DECLARED AS VAR
*
*           PULL WITHOUT PUSH
*

* E198    :  SYNTAX ERROR IN VARIABLE SYMBOL DECLARATION
*
*
*

* E199    :  SYNTAX ERROR IN VARIABLE DECLARATION
*
*
*
```

* --- POOR OR FAULTY ENCODER SIGNALS

* E210 : POOR OR FAULTY POSITION SIGNALS ON AXIS 0
*
*
*

* E211 : POOR OR FAULTY POSITION SIGNALS ON AXIS 1
*
*
*

* E212 : POOR OR FAULTY POSITION SIGNALS ON AXIS 2
*
*
*

* E213 :
*
*
*

* E214 :
*
*
*

* E215 :
*
*
*

* E216 :
*
*
*

* E217 :
*
*
*

* E218 :
*
*
*

* E219 :
*
*
*

NOTES

NUM in the Great Britain and in the United States

List updated 28-04-92

		Address	Telephone Telex (Tx)	Fax
Great Britain	❐ ■	NUM SERVOMAC Ltd - Coventry	(203) 69.25.25	(203) 69.30.34
United States	❐ ■	NUM Corp. - Naperville (IL.)	(708) 505.77.22	(708) 505.77.54

NUM in the world

Head Office				
France	❐ ■	NUM SA 21, Avenue du Maréchal Foch BP 68 - 95101 Argenteuil Cedex	(1) 34.23.66.66 Tx : 609 611	(1) 34.23.65.49
Subsidiaries				
Germany	❐ ■	NUM-GÜTTINGER GmbH - Stuttgart/Nellingen	(711) 34.80.60	(711) 348.06.10
	❐	NUM-GÜTTINGER - Verkaufsbüro West-Mechernich	(2443) 59.59	(2443) 62.43
	❐	NUM-GÜTTINGER - Verkaufsbüro Nord-Löhne 2	(5732) 820.07	(5732) 826.43
	❐ ■	NUM-GÜTTINGER - Technisches Büro-Ratingen	(2102) 830.97	(2102) 84.37.51
	❐ ■	NUM-GÜTTINGER - Technisches Büro-Chemnitz	(722) 64.31.47 Tx : 777 61, 777 63	(722) 64.31.50
Italy	❐ ■	NUM Spa - Bologna/Zola Predosa	(51) 75.41.18	(51) 75.40.82
	❐ ■	NUM Spa - Milano/Agrate Brianza	(39) 605.73.26	(39) 65.44.68
	❐ ■	NUM Spa - Torino/Rivoli	(11) 955.03.59	(11) 955.03.62
	❐ ■	NUM SERVOMAC (Speed change drive unit) - Milano	(2) 27.00.22.93	(2) 27.00.21.53
Spain	❐ ■	TELENUM SA - Itziar - Deba (Guipuzcoa)	(43) 19.92.55	(43) 19.91.05
	❐ ■	TELENUM SA - Barcelona/Castelldefels	(3) 636.13.52	(3) 636.29.77
	❐	TELEMECANICA ELECTRICA ESPAÑOLA SA Madrid	(1) 695.71.00 Tx : 22702	(1) 682.08.74
Sweden	❐ ■	NUM NORDEN AB - Västerås	(21) 13.11.31	(21) 13.12.30
Switzerland	❐ ■	NUM-GÜTTINGER AG - Teufen	(71) 33.04.11 Tx : 883 975	(71) 33.35.87
	❐ ■	NUM-GÜTTINGER SA - Bienne	(32) 23.53.73 Tx : 34 333	(32) 23.55.25
Representatives				
Argentina	❐ ■	FASTER SA - Buenos Aires	(1) 766.22.26 (1) 765.63.42	(1) 763.63.92
Belgium	❐ ■	CARON VECTOR SA - Bruxelles	(10) 23.13.11 Tx : 59 509	(10) 23.13.36
CIS	■	KHARKOV TECHNICAL MAINTENANCE CENTRE - Kharkov/Ukraine	(0572) 23.64.40 (0572) 23.64.41 Tx : 125 649 (National) Tx : 115 147 (International)	
Taiwan	❐	TECHRIDGE INDUSTRIAL Co Ltd - Taipei	(2) 731.59.07	(2) 775.37.19
Tunisia	■	SIME - Tunis	(1) 78.57.13	(1) 78.95.88

❐ - Distribution

■ - After-sales Service

NUM in the world

List updated 28-04-92

Partners or Representatives		Address	Telephone Telex (Tx)	Fax
China	■	TETS - BEIJING TECHNICAL & SERVICE CENTER - Beijing	(1) 408.22.11 Tx : 210 383	(1) 408.18.62
Colombia	■	TELEMECANIQUE DE COLOMBIA - Bogota	(1) 413.91.81 Tx : 43 191	(1) 413.90.12
Finland	❒	NUCOS OY - Helsinki	(0) 294.35.90	(0) 294.75.90
	❒	NUCOS OY - Tampere	(31) 79.09.29	(31) 79.08.37
Hong-Kong	❒	TELEMECANIQUE ASIA PACIFIC Ltd - Hong-Kong	(5) 65.06.21 Tx : 86 508	(8) 11.10.29
India	■	THE PREMIER AUTOMOBILES Ltd - Pune	(212) 77.51.61 Tx : 146 244	(212) 77.61.84
Japan	■	TELEMECANIQUE JAPAN Ltd - Tokyo	(3) 35.88.86.71	(3) 35.88.89.02
Korea	■	TELKO Ltd - Seoul	(2) 679.25.26	(2) 682.81.07
Mexico	■	TELEMECANIQUE MEXICO SA - Naucalpan/Mexico	(5) 358.86.33 Tx : 176 13 79	request transmission by fax
South Africa	■	MACHINE TOOL TECHNOLOGIES - Isando	(11) 823.17.55 (11) 823.17.56	(11) 823.17.05
Taiwan	■ ❒	NUM TELEMECANIQUE TAIWAN - Taichung	(4) 328.20.81 (4) 328.20.82	(4) 328.20.83

❒ - Distribution

■ - After-sales Service