10 Series CNC
Programming Manual

Code: 45004457K
Rev. 17

PUBLICATION ISSUED BY:

OSAI S.p.A.
Via Torino, 14 - 10010 Barone Canavese (TO) – Italy

Phone: +39-0119899711

Web: www.osai.it

e-mail: sales@osai.it
service@osai.it

Edition: December 2006

IMPORTANT USER INFORMATION

This document has been prepared in order to be used by OSAI. It describes the latest release of the product.
OSAI reserves the right to modify and improve the product described by this document at any time and without prior notice.
Actual application of this product is up to the user. In no event will OSAI be responsible or liable for indirect or consequential damages that may result from installation or use of the equipment described in this text.

SUMMARY OF CHANGES

General

This publication is issued with reference to Software Release 7.6 (E69).

PAGE	UPDATING TYPE
INDEX	Updated
CHAPTER 1	
p. 1	Added description
p. 11	Added code G98
CHAPTER 2	
p. 13	Changed paragraph “CRT – Circular interpolation speed reduction threshold. CRK – Circular interpolation speed reduction constants”
p. 44	Added new paragraph “AXF – Definition of axes with dynamic following function”
p. 56	Added description
p. 57	Added description
pp. 58-59	Added description
p. 61	Changed paragraph “MOV – Movement modalities”
p. 70	Changed paragraph “JRK – Jerk Constant”
p. 135	Added description in table of “operational limit modes”
p. 161	Added new paragraph “TCP – Tool Center Point for machines with prismatic head”
CHAPTER 3	
p. 8-9	Added description
CHAPTER 4	
p. 5	Changed description in paragraph “Notes on using cutter diameter compensation”
p. 16	Changed paragraph “TPO – Tool path optimisation in G41/G42”
p. 26	Added new paragraph “TPA – Threshold angle for TPO”
p. 34	Added new paragraph “UVW – Definition of axes for paraxial compensation”
CHAPTER 12	
p. 23	Changed description
CHAPTER 14	
p. 11	Added description
pp. 12-13	Added description
pp. 18-24	Added description
CHAPTER 16	
p. 9	Corrected description n_pass

CHAPTER 17	
	Added new chapter “Filters”
APP. B	
p. 1	Added description for NC008
p. 4	Changed description for NC042
APP. C	
p. 5	Added new paragraph “Errors in multiprocess management”

PREFACE

This manual describes the procedures used for writing part programs with the 10 Series CNC system. It provides programmers with all the information they need for creating machine control programs.

REFERENCES

For further information:

- 10 Series CNC - AMP Software Characterization Manual
- 10 Series CNC - User Guide

The chapters in this manual are organised in sections. They describe the language elements (commands and functions) used for managing a specific task, e.g. axis programming, tool programming, probe management. Programming examples have been introduced in the command description.

SUMMARY

1. **Programming with 10 Series System**

   This chapter contains the general programming rules of the International Standards Organization (ISO) standard. The chapter also provides an overview of the programming environment and a summary of the most used codes.

2. **Programming the Axes**

   This chapter describes axis programming. The G codes and extended commands involved in this activity are provided with their characteristics. Several examples complete the command description and give suggestions for programming the major types of movements.

3. **Programming Tools and tool offsets**

   This chapter describes tool programming and provides the functions and instructions used in tool operation.

4. **Cutter Diameter Compensation**

   This chapter describes cutter compensation. T functions and G codes used in tool compensation are provided with characteristics and several examples.

5. **Programming the Spindle**

   This chapter describes spindle programming. The G codes and extended commands involved in this activity are provided with their characteristics. Several examples complete the command description and give hints for solving the main cases of spindle programming.

6. **Miscellaneous Functions**

   This chapter describes miscellaneous functions and provides a list of M functions with their meaning and characteristics.

7. **Parametric Programming**

   This chapter deals with special programming applications that use local and system variables.

8. **Canned Cycles**

   This chapter provides a description of the canned cycles available with the control. The G codes and extended commands used in this activity are provided with their characteristics. Several examples complete the command description.

9. **Paramacros**

   This chapter describes how paramacros can be used in programs.

10. **Probing Cycles**

    This chapter provides a description of the probing cycles available with the control. The G codes and extended commands involved in probe management are provided complete with examples.

11. **Managing the Screen**

    This chapter discusses the commands used to handle the system screen from a part programs. Examples are given to complete the command description.

12. **Modifying the Program Execution Sequence**

    This chapter contains the commands used for modifying the sequence of execution of a part program. It describes commands for branching, repeating blocks and executing subprograms, as well as commands for putting the part program on hold and releasing it.

13. **High Speed Machining**

    This chapter describes the high-speed milling features on machine tools with 3 axes.

14. **Multiprocess management commands**

    This chapter shows 10 Series CNC's multi process potentials.

15.High level geometric programming (GTL)

This chapter discusses the set of programming instructions available with the GTL utility.

16.Working Cycles for Turning Systems

This chapter provides the instructions for programming macro-cycles of rough-shaping, threading and groove cutting.

17.Filters

This chapter describes the various types of filters that can be configured and hence applied in OSAI control units, designed to improve machine tool performances from the geometric and dynamic points of view and hence the finishing quality of the parts produced.

A. Characters and Commands

Appendix A provides a summary of all the characters allowed in the system and gives lists of G codes, mathematical functions and extended commands.

B. Error Messages

Appendix B provides a list of all the error messages that can occur during programming..

C. Error management

COMMANDS

Commands are dealt with in the chapters that describe the specific task. A common structure has been adopted in the command description. For each command, the following information is provided:

- Command name
- Command function
- Command syntax
- Parameters
- Characteristics and notes
- Examples

Where possible, examples consist of a portion of program and a diagram that shows how the commands in that portion work.

Syntax conventions

Use these conventions with the commands:

SYMBOL	MEANING
[ ]	Brackets enclose optional entries. Do not enter the brackets.
{ }	Braces enclose entries which may be repeated more than once. This could also be described as a series of alternative entries, i.e. only one of these may be entered. Alternative entries are separated by a (**|**). Do not enter the braces in the command itself.
|	A vertical bar separates alternative entries. Do not enter the bar.

Key-words are written in **bold**. They must be entered exactly as they are represented in the syntax description.
Parameters that must be passed with commands are indicated by a mnemonic written in italics. Appropriate values must be entered in place of the mnemonic. Leading zeros can be omitted. For example, you can program G00 as G, G01 as G1.

Example:
(SCF,[*value***])**
SCF, the comma and parenthesis are key-words and must be written as described. *value* is a parameter name and must be replaced by an appropriate value. The brackets indicate that *value* is an optional value.

Warnings

For correct control operation, it is important to follow the information given in this manual. Take particular care with topics bearing one of the mentions: WARNING, CAUTION or IMPORTANT, which indicate the following types of information:

WARNING	Draws attention to facts or circumstances that may cause damage to the control, to the machine or to operators.
CAUTION	Indicates information to be followed in order to avoid damage to equipment in general.
IMPORTANT	Indicates information that must be followed carefully in order to ensure full success of the application.

Terminology

Some terms appearing throughout the manual are explained below.

Control	Refers to the 10 Series numerical control unit comprising front panel unit and basic unit.
Front Panel	Is the interface module between machine and operator; it has a monitor on which messages are output and a keyboard to input the data. It is connected to the basic unit.
Basic Unit	Is the hardware-software unit handling all the machine functions. It is connected to the front panel and to the machine tool.

END OF PREFACE

INDEX

PROGRAMMING WITH 10 SERIES SYSTEMS

PROGRAMMING THE AXES

PROGRAMMING TOOLS AND TOOL OFFSETS

CUTTER DIAMETER COMPENSATION

PROGRAMMING THE SPINDLE

MISCELLANEOUS FUNCTIONS

PARAMETRIC PROGRAMMING

CANNED CYCLES

PARAMACRO

PROBING CYCLES

MANAGING THE SCREEN

MODIFYING THE PROGRAM EXECUTION SEQUENCE

HIGH SPEED MACHINING

MULTIPROCESS MANAGEMENT COMMANDS

HIGH LEVEL GEOMETRIC PROGRAMMING (GTL)

WORKING CYCLES FOR TURNING SYSTEMS

FILTERS

CHARACTERS AND COMMANDS

ERROR MESSAGES

ERROR MANAGEMENT

END OF INDEX

PROGRAMMING WITH 10 SERIES SYSTEMS

10 Series part programs are written with a specific language defined by the ISO standard. This chapter describes the language elements and discusses programming techniques and rules.

THE PROGRAM FILES

The 10 Series part programs are stored in files which may be identified with 10 SERIES names or with DOS names.

- 10 SERIES names are a maximum of 48 characters in length; they identify the programs stored in the logic directories configured on the machine.
  Logic directories are configured during the installation stage (PPDIR config - human interface menu in AMP characterization).
- DOS names are a maximum of 8 characters in length, plus an extension and path where applicable; they identify files resident in DOS type directories.
- Using the Windows Editors, remember to give the “enter” command on the last line entered in the program. Quitting the program without this command might cause errors during program execution.

Mixed management of part programs is not allowed; in fact if a program is activated after being called by a DOS type name, all it subroutines must be identified with DOS names.
Similarly, programs with 10 SERIES names can use only subroutines identified in the same way.

NOTE:
Part programs can also be resident on remote devices, defined in advance through the triliteral GDV (see chap. 12).

Program Components

- ***Address***

  An address is a letter that identifies the type of instruction. For example, these are addresses:

  G, X, Y, F

- ***Word***

  A word is an address followed by a numerical value. For example, these are words:

  G1 X50.5 Z-3.15 F200 T1.1

  When you assign a numeric value to a word, no zeroes must preceed or follow the value. Insert decimal values after the decimal point.

- ***Block***

  A program block comprises a set of words that identify an operation or a series of operations to be performed. The maximum length of a block is 126 characters.
  A technological program is a sequence of blocks that describe a machining operation.

  Each block must end with: <CR> <LF>.

Blocks

Blocks may include one or several fields.
When several fields are used in the same block, they must appear in the order shown in the following table:

block delete	label	sequence number	synchronisation asynchronisation	words codes
/	LABEL	NUMBER	# or &	ALL ALLOWED CHARACTERS

- ***Comment blocks***

  It can be inserted in any position within the current block. Any character after ";" is considered as a comment.

- ***Block delete***

  The block delete field is optional. It allows the operator to choose whether to execute program blocks that begin with the "/" character that are called slashed blocks.

  Example:
  /N100 G00 X100

  The block shown in the example can be enabled or disabled using the PROGRAM SET UP softkey, or typing the three-letter code DSB on the keyboard.

- ***Label***

  The label field is optional. It allows the programmer to assign a symbolic name to a block. A label can have up to six alphanumeric characters which must be between quotes. In case of a slashed block, the label must be inserted after the slash.

  Example:
  "START"
  /"END"

  When a label field is used in a 'GTO' command, the label defines the block that the control should jump to.

- ***Sequence number***

  The "sequence number" field is optional. It allows the programmer to number each program block. A sequence number begins with the letter N and is followed by up to six digits (N0-N999999).

  The sequence number must appear in front of the first operand and after the label.

  Example:
  N125 X0
  "START" N125 X0
  "END" N125 X0

- ***Synchronisation/asynchronisation***

  Characters & and # are used to override the default synchronisation/asynchronisation status. For further information on synchronisation, see "Synchronisation and Program Execution".

  Example:
  #(GTO,START, @PL1=1)

Block Types

Four types of blocks can be used in a part program:

- Comment blocks
- Motion blocks
- Assignment blocks
- Three-letter command blocks

- ***Comment blocks***

  A comment block allows the programmer to insert free sentences in the program. These sentences may describe the function to be executed or provide other pieces of information that make the program more understandable and documented.
  A comment block does not produce messages for the operator. The control ignores a comment block during execution of the program.
  The first character of a comment block must be a semicolon (;). The rest of the comment block is a sequence of alphanumeric characters. For example:

```
;THIS IS AN EXAMPLE OF COMMENT BLOCK
```

  A comment can be inserted not only in a single block, but also in other types of blocks after the character ";".All characters after a ; considered as a comment. For example:

```
G1 X100 Y50 ; Motion block
E1=10 ; Local variable E
(ROT,45) ; Rotation command
```

♦ ***Motion blocks***

Motion blocks conform to ISO and ASCII standards for programming blocks. There is no particular order for programming the components of a motion block.

Example:

```
G1 X500 Y20 F200
```

♦ ***Assignment blocks***

Assignment blocks are used to write variables' values directly from the program. Several types of assignments are possible as shown in the following table:

TYPE OF ASSIGNMENT	EXAMPLE
Simple assignment	E10=123.567
Multiple assignment	E1=10, 15.5, 123.467 In multiple assignments values are loaded as follows: 10 to E1 15.5 to E2 123.467 to E3
Math expression assignment	E20=(E10+125*SQR(E23))
System number	SN=1.5

♦ ***Three-letter command blocks***

Three-letter command blocks define an operation with a three-letter instruction in conformity with the RS-447 standard. For example:

```
(ROT,45)
(DIS,"message text")
```

For the sake of compatibility between 10 Series and Series 8600 certain commands may be programmed with either of the following three-letter codes.

UGS	UCG
CGS	CLG
DGS	DCG
RQT	RQU
DPA	DSA
PAE	ASC
PAD	DSC
DPP	DPT
IPB	DTL
ROT	URT
SOL	DLO
UTO	UOT
TOU	TOF

Programmable Functions

♦ ***Axis coordinates***

Axis coordinates can be named with letters ABCUVWXYZPQD (according to the configuration set in AMP) and can be programmed in the following ranges:

-99999.99999	-0.00001	mm/inch
+0.00001	+99999.99999	mm/inch

NOTE:
It is impossible to program coordinates in the ±0.00001 range because 0.00001 is the minimum value accepted by the control.

♦ ***R coordinate***

In a circular interpolation (G02 G03) R represents the radius of the circle.
In a standard canned cycle (G81-G89), the R coordinate defines the initial position value and retract value. This function is programmable in the following ranges:

-99999.99999	-0.00001	mm/inch
+0.00001	+99999.99999	mm/inch

NOTE:
It is impossible to program values in the ±0.00001 range because 0.00001 is the minimum value accepted by the control.
In a threading block (G33), the R coordinate represents the offset from the zero angular position of the spindle for multi-start threads.

♦ ***I J coordinates***

In circular interpolation (G02-G03), I and J specify the coordinates of the center of an arc. I specifies the abscissa (typically X) and J the ordinate of the center (typically Y). I and J always specify the center coordinates regardless of the active interpolation plane.
This function is programmable in the following ranges:

-99999.99999	-0.00001	mm/inch
+0.00001	+99999.99999	mm/inch

NOTE:
It is impossible to program values in the ±0.00001 range because 0.00001 is the minimum value accepted by the control.

When the values of the corresponding axis are expressed in diametrical units (according to the configuration set in AMP), the values of the center coordinates (I and J) are also expressed in diametrical units.

I and J coordinates are also used in the deep hole drilling cycle (G83).
In a threading block (G33), the I address defines the pitch variation for variable pitch threads:

I+	Increasing pitch
I-	Decreasing pitch

♦ ***K function***

In the deep hole drilling cycle (G83), K defines the incremental value to be applied to the minimum depth value (J) in order to reduce the initial pitch depth (I).
This function is programmable in the following ranges:

-99999.99999	-0.00001	mm/inch
+0.00001	+99999.99999	mm/inch

NOTE:
It is impossible to program values in the ±0.00001 range because 0.00001 is the minimum value accepted by the control.

In a threading block (G33) or a tapping cycle (G84), K defines the thread pitch. In helical interpolation (G02-G03), K defines the helix pitch.

- ***F and t function***

The F function defines the axes feedrate. This function is programmable in the following range:

+0.00001	+99999.99999	mm/inch

In G94, F function defines the feedrate in millimetres per minute (G71) or inches per minute (G70).

A "t" value can be programmed in a block to specify the time in seconds needed to complete the move defined in the block. In this case the block feedrate will be:

$$F = \frac{\text{total distance}}{\text{time}} * 60$$

A "t" value is valid only in the block in which it is programmed.
In G93, the F function defines the inverse of the necessary time in minutes to complete the movement:

$$F = \frac{\text{speed}}{\text{total distance}} = 1/t \text{ (minutes)}$$

The F function is mandatory in the blocks when G93 is active and only affects that block.
In G95, F specifies the axes feedrate in millimetres per revolution (G71) or inches per revolution of the spindle (G70).

- ***a Function***

The a function defines the acceleration to use on the part program block and may be programmed in the range:

+0.00001	+99999.99999	mm/sec^2 or inches/sec^2

The a function is considered in mm/sec^2 in presence of G71 and in inches/sec^2 in presence of G70. This function is active only in the block it is programmed in and is in any case limited to the acceleration on the profile as calculated by the system in function of the accelerations configured.

- ***M function***

The M address can activate various machine operations. The programmable range goes from 0 to 999. See Chapter 6 for further information about these functions.

♦ *S function*

The S function specifies the spindle rotation speed. It is programmable in the following range:

+0.001	999999.999

rpm/fpm

In G97, the S function defines spindle rotation speed expressed in revolutions per minute.

In G96, the S function defines the cutting surface speed expressed in metres per minute (G71) or feet per minute (G70). The above cutting speed remains constant on the surface.

Refer to Chapter 5 for further information about S function programming.

♦ *T function*

The T function defines the tool and tool offset needed for machining. It is programmable in the 0.0 to 999999999999.300 range. The 12 digits on the left of the decimal point represent the tool identifier code and the three digits on the right represent the tool offset number.

Chapter 3 provides a detailed description of T functions.

IMPORTANT

M, S and T functions vary according to their characterisation in AMP.
From SW release 3.1 it is possible for the system to execute these functions inside a continuous move (G27-G28).
When planning an application the manufacturer must:

- configure the desired function as "ALLOWED IN CONTINUOUS" in AMP.
- write a machine logic to handle such a function.

In turn, the programmer must remember that these functions produce different effects depending on how they are programmed:

- in **continuous mode** a function configured as "ALLOWED IN CONTINUOUS" will be executed in the sequence in which it has been programmed. In order not to lock the program the function will be executed in "NO WAIT" mode.
- in **point-to-point mode** a function configured as "ALLOWED IN CONTINUOUS" will be executed in standard mode.

♦ *h functions*

h functions permit to alter an offset during both continuous and point to point moves.
An h function must be programmed by itself in a block. Its value may range from 0 through 300 and may be either an integer or an E variable.

♦ *G functions*

G codes program machining preparatory functions for machining. The following section deal with this codes.

G Codes

This section shows how to write preparatory G codes in part program blocks. A preparatory G code is identified by the G address followed by one or two digits (G00-G99). At present, only some of the 100 possible G codes are available.

Paramacro subroutines can be called with a three-digit G code. This class of G codes is described in Chapter 9. Three-digit G codes are classified as follows:

G100 - G299	Reserved
G300 - G699	Non modal paramacro range
G700 - G998	Modal paramacro range
G999	Reset modal paramacro

The G code must be programmed after the sequence number (if defined) and before any other operand in the block. For example:

N100 G01 X0 - operand

It is possible to program several G codes in the same block, provided they are compatible with each other. The table that follows defines compatibility between G codes. Zero indicates that the G codes are compatible and can be programmed in the same block; 1 means that the G codes are not compatible and cannot be programmed in the same block without generating an error.

Compatible G Codes

G	00	01	02 03	33	81 89	80	72 73 74	93 94 95	96 97	41 42	40	27 28	29	04	09	90 91	79	70 71	16 17 18 19	92 99 98	20	21	60 61
G00	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G01	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G02	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G03	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G04	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	1
G09	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1
G16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G20	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	1	1	1	1	1
G21	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	1	1	1	1	1
G27	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1
G28	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1
G29	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1
G33	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1
G40	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
G41	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	0	1	1	1	0	1
G42	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	0	1	1	1	0	1
G60	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G61	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G70	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1
G71	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1
G72	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G74	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G79	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1	1	0	1	1	1	1	1
G80	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	1	0	1	1	1	1	1
G81	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1	1	1	1	1
G82	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G83	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G84	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G85	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G86	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G89	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1
G90	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1
G91	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1
G92	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G93	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G94	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G95	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G96	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G97	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
G98	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G99	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

NOTE:
0 means compatible G codes
1 means incompatible G codes

The following table gives a summary of the G codes available in the control. This default configuration can be modified through the AMP utility.

G code summary

CODE	GROUP	MODAL	DESCRIPTION	POWER UP	
				MILL	GRINDING
G00	a	yes	Rapid axes positioning	yes	yes
G01	a	yes	Linear interpolation	no	no
G02	a	yes	Circular interpolation CW	no	no
G03	a	yes	Circular interpolation CCW	no	no
G33	a	yes	Constant or variable pitch thread	no	no
G16	b	yes	Circular interpolation and cutter diameter compensation on a defined plane	no	no
G17	b	yes	Circular interpolation and cutter diameter compensation on 1st-2nd axes plane	yes	no
G18	b	yes	Circular interpolation and cutter diameter compensation on 3rd-1st axes plane	no	yes
G19	b	yes	Circular interpolation and cutter diameter compensation on 2nd-3rd axes plane	no	no
G27	c	yes	Continuous sequence operation with automatic speed reduction on corners	yes	yes
G28	c	yes	Continuous sequence operation without speed reduction on corners	no	no
G29	c	yes	Point-to-point operation	no	no
G92	d	no	Axis presetting without mirror	no	no
G98	d	no	Axis presetting with mirror	no	no
G99	d	yes	Delete G92	yes	yes
G40	e	yes	Cutter diameter compensation disable	yes	yes
G41	e	yes	Cutter diameter compensation-tool left	no	no
G42	e	yes	Cutter diameter compensation-tool right	no	no
G20		yes	Closes GTL profile		
G21		yes	Opens GTL profile		
G60		yes	Closes the HSM profile	no	no
G61		yes	Opens the HSM profile	no	no
G62		no	Splits the HSM profile in two with continuity	no	no
G63		no	Splits the HSM profile in tw with link	no	no
G66		no	Splits the HSM profile in two with edge	no	no
G67		no	Splits the HSM profile in two with reduced speed on edge	no	no

CODE	GROUP	MODAL	DESCRIPTION	POWER UP MILL	 GRINDING
G70	f	yes	Programming in inches	no	no
G71	f	yes	Programming in millimetres	yes	yes
G80	g	yes	Disable canned cycles	yes	yes
G81	g	yes	Drilling cycle	no	no
G82	g	yes	Spot-facing cycle	no	no
G83	g	yes	Deep hole drilling cycle	no	no
G84	g	yes	Tapping cycle	no	no
G85	g	yes	Reaming cycle	no	no
G86	g	yes	Boring cycle	no	no
G89	g	yes	Boring cycle with dwell	no	no
G90	h	yes	Absolute programming	yes	yes
G91	h	yes	Incremental programming	no	no
G79	i	no	Programming referred to axis home switch	no	no
G04	j	no	Dwell at end of block	no	no
G09	j	no	Deceleration at end of block	no	no
G72	k	no	Point probing with probe tip radius compensation	no	no
G73	k	no	Hole probing with probe tip radius compensation	no	no
G74	k	no	Probing for theoretical deviation from a point without probe tip radius compensation	no	no
G93	l	yes	Inverse time (V/D) feedrate programming mode	no	no
G94	l	yes	Feedrate programming in ipm or mmpm	yes	no
G95	l	yes	Feedrate programming in ipr or mmpr	no	yes
G96	m	yes	Constant surface speed (feet per minute or metres per minute)	no	yes
G97	m	yes	Spindle speed programming in rpm	yes	no

SYNCHRONISATION AND PROGRAM EXECUTION

The terms "synchronised" and "asynchronised" apply only to part program blocks that do not imply a movement, that is, assignment or calculation blocks. A motion block is any block containing axes motion together with other actions:

- Axis moves
- M codes
- S codes
- T codes

A synchronisation block is taken into consideration and executed only after the motion block that precedes it in the program is completed, that is after the axis move has been executed.

On there other hand, a non-synchronised block is executed as soon as it is read by the part program interpreter, i.e. when perhaps the previous move is still in progress.

The advantage of asynchronous block execution is that variable assignments and complex calculations can be made between moves. This allows to reduce waiting time between two motion blocks caused by calculations.

Default Synchronisation

At power up, the following commands and codes are automatically synchronised:

- UDA, SCF, RQO, IPB, DLY, WOS, WAI, SND, GTA, REL, UPR, TCP, UVP, UVC
- G16, G17, G18, G19, G72, G73, G74

All the other commands are not synchronised.

This default assignment can be changed. This means that the commands that are synchronised by default at power-up can become asynchronous and that the commands that are not synchronised by default at power-up can become synchronous. The next section explains how to override default synchronisation.

NOTE:
Default synchronisation cannot be modified for GTA, UPR, TCP, UVP, and UVC instructions.

Overriding Default Synchronisation

Under certain circumstances, the part program may request to modify the default synchronisation.

If the command is synchronised by default and the programmer wants it to be executed by the interpreter as soon as it is read (asynchronous operation), an "&" must be programmed in the first position of the block, immediately after the "n" number.

If the command is asynchronous and you wish to activate synchronous operation, the first character in the block must be #.

Both # and & are active only in the block where they are programmed.

WARNING

To avoid possible damage to the workpiece, note that programming synchronised blocks between contouring blocks clears the motion buffer at each synchronised block. This will result in dwells while the buffer is reloaded and all the calculations are performed.

Part Program Interpreter

When the system reads a part program block it executes various activities, depending on the type of block:

- A motion block will be loaded in the motion buffer queue. If the move is defined by a variable, the stored move values stored are those of the variable. The buffer size is configurable from 2 to 128 blocks through AMP.
- An asynchronous assign or calculation block will be executed.

Three factors cause the part program interpreter to stop reading blocks:

- The motion buffer is full. When the active motion block is completed, the interpreter will read another motion block and load it in the buffer queue.
- A non-motion block that contains a synchronised command or a code that forces synchronisation is read. The interpreter does not start again until the last loaded motion block is completed. At this point the block calling for synchronisation is executed and the interpreter starts reading the following blocks.
- Error conditions

Sequence of execution

1. Diameter axes
2. Scale factors (SCF)
3. Measuring units (G70 G71)
4. Paraxial compensation (u v w)
5. Inch/metric programming (G90 G91)
6. Mirror machining (MIR)
7. Plane rotation (ROT)
8. Origins (UAO UTO UIO G92)

Programming restrictions for long real (double) formats

The following restrictions apply to long real programming:

- Max. 15 numbers in total
- Max. 12 integer digits
- Max. 9 decimal digits

The system will display an error if more than 12 integer digits are programmed.

If more than 9 decimal numbers are programmed, the system does not display any error but cuts off the programmed number at the last allowed digit.

END OF CHAPTER

PROGRAMMING THE AXES

AXIS MOTION CODES

Defining Axis Motion

In this manual axes motion directions are defined in compliance with EIA standard RS-267. By convention, we always assume that the tool moves towards the part, no matter whether the tool moves towards the part or the part moves towards the tool in the actual process.

Basic movements can be defined with the motion G codes listed in the following table:

G CODE	FUNCTION
G00	Rapid axes positioning
G01	Linear interpolation
G02	Circular interpolation clockwise
G03	Circular interpolation counter clockwise
G33	Constant or variable pitch threading

G00 - Rapid Axes Positioning

G00 defines a linear movement at rapid feedrate that is simultaneous and coordinated for all the axes programmed in the block.

Syntax

G00 [*G-codes*] [*axes*] [*offset*] [**F**..] [**a**] [*auxiliary*]

where:

G-codes	Other G codes that are compatible with G00 (See "Compatible G codes" table in Chapter 1).
axes	Axis name followed by a numerical value. The numerical value can be programmed directly with a decimal value or indirectly with an E parameter. Up to nine axes can be written in a block.
offset	Offset factors on the profile. For the X, Y, Z axes these factors are entered with u, v, and w respectively. See "Paraxial compensation" in Chapter 4 for further information.
F	Feedrate for coordinated moves. It is given with the F address followed by the feedrate value. This parameter does not affect the move of the axes programmed in the G00 block, but is retained for subsequent feedrate moves. The rapid feedrate forced by G00 is a velocity along the vector of the axes programmed in the block. The maximum rapid feedrate is defined during characterisation with the AMP utility.
a	Acceleration to be used on the profile.
auxiliary	Programmable M, S, and T auxiliary functions. Up to four M functions, one S (spindle speed) and one T (tool selection) can be programmed in the block.

G01 - Linear Interpolation

G01 defines a linear move at machining feedrate that is simultaneous and coordinated on all the axes programmed in the block.

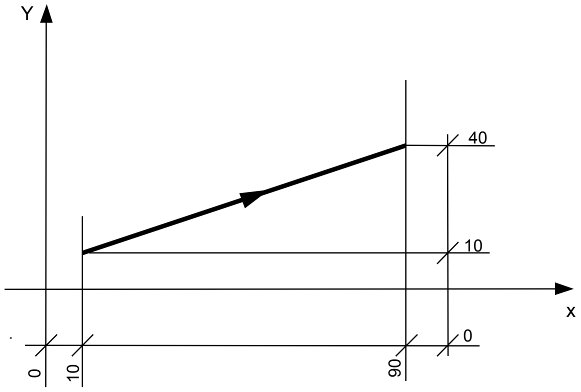
Syntax

G01 [*G-codes*] [*axes*] [*offset*] [**F**..] [**a**] [*auxiliary*]

where:

G-codes	Other G codes that are compatible with G01 (See "Compatible G codes" table in Chapter 1).
axes	Axis name followed by a numerical value. The numerical value can be programmed directly with a decimal value or indirectly with an E parameter. Up to nine axes can be written in a block.
offset	Offset factors on the profile. These factors are entered for the X, Y, Z axes with the characters u, v, w respectively. See "Paraxial compensation" in Chapter 4 for further information.
F	Feedrate used for the move. It is given with the F address followed by the feedrate value. If omitted, the system will use the previously programmed feedrate. If no feedrate has been programmed the control will generate an error.
a	Acceleration to be used on the profile.
auxiliary	Programmable M, S, T auxiliary functions. Up to four M functions, one S (spindle speed) and one T (tool selection) can be programmed in the block.

Example:
This example shows how to program a G01 code.



Program:

```
N60 (UGS,X,-10,100,Y,-10,50)
N70 G0 X10 Y10
N80 G01 X90 Y40 F200
```

G02 G03 - Circular Interpolation

These codes define the following circular movements:

G02	Circular interpolation clockwise (CW)
G03	Circular interpolation counter clockwise (CCW)

The circular move is performed at machining feedrate and is coordinated and simultaneous with all the axes programmed in the block.

Syntax

G02 [*G-codes*] [*axes*] **I**.. **J**.. [**F**..] [**a**] [*auxiliary*]
or
G02 [*G-codes*] [*axes*] **R**.. [**F**..] [**a**] [*auxiliary*]

G03 [*G-codes*] [*axes*] **I**.. **J**.. [**F**..] [**a**] [*auxiliary*]
or
G03 [*G-codes*] [*axes*] **R**.. [**F**..] [**a**] [*auxiliary*]

where:

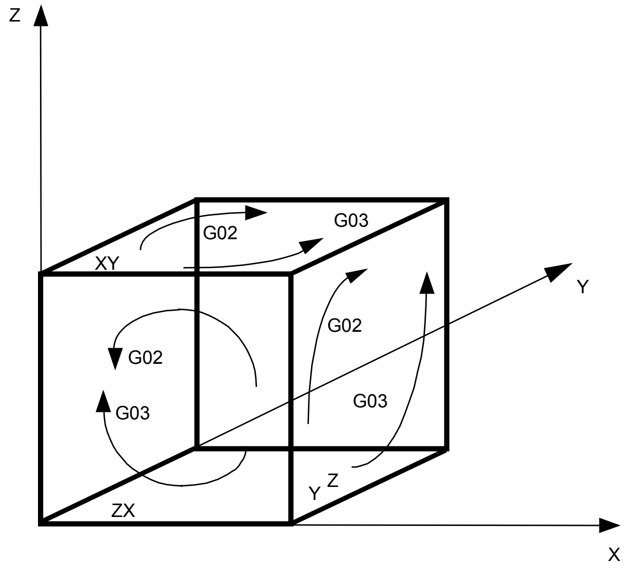
G-codes	Other G codes that are compatible with G02 and G03 (See "Compatible G codes" table in Chapter 1).
axes	Axis name followed by a numerical value programmed directly with a decimal value or indirectly with an E parameter. If axes are not programmed in the block, the move is a complete circle in the active interpolation plane.
I	Abscissa of the circle centre. This is a value in millimetres that can be programmed directly or indirectly with an E parameter. The abscissa is expressed as a diameter unit when the corresponding axis is a diameter axis. No matter what interpolation plane you are using, the symbol for the abscissa is always **I**.
J	Ordinate of the circle centre. This is a value in millimetres that can be programmed directly or indirectly with an E parameter. The ordinate is expressed as a diameter unit when the corresponding axis is a diameter axis. No matter what interpolation plane you are using, the symbol for the ordinate is always **J**. **NOTE:** The parameter **R** cannot be used for arcs of 360 degrees..

R	Circle radius alternative to the I and J coordinates. If the arc of a circle is less than or equal to 180 degrees, the radius must be programmed with positive sign; if the arc of a circle is greater than 180 degrees the radius must be programmed with negative sign. **NOTA:** R is not allowed with arc of 360 degrees.
F	Feedrate used for the move. It is given with the F address followed by the feedrate value. If omitted, the system will use the programmed value. If no feedrate has been programmed an error will occur.
a	Acceleration to be used on the profile.
auxiliary	Programmable auxiliary functions M, S, T. Up to four M functions, one S (spindle speed) and one T (tool selection) can be programmed in the block.

Characteristics:

The maximum programmable arc is 360 degrees, i.e. a full circle. Before programming a circular interpolation block, the interpolation plane must be defined with G16, G17, G18, G19. G17 is automatically active after power up.
The coordinates of the start point (determined from the previous block), the end point and the centre of the move must be calculated so that the difference between start and end radius is less than the default value (0.01 mm or 0.00039 inches). If this difference is equal or greater than the default value, the control displays an error message and the circular move is not performed.

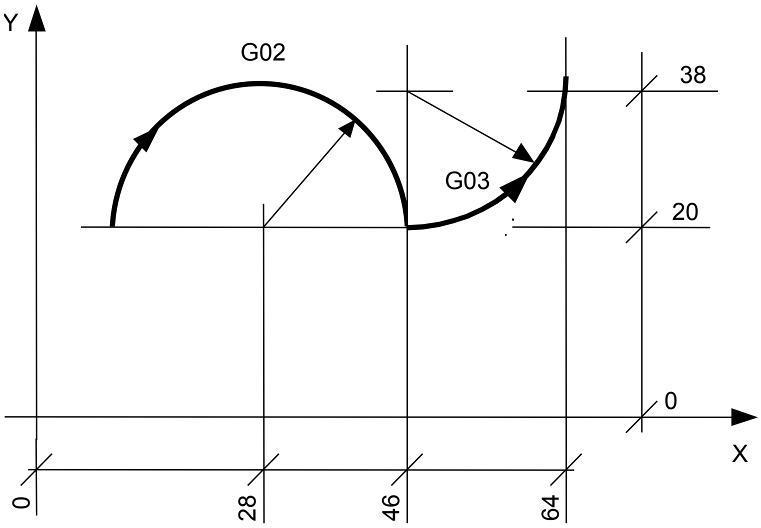
Incremental programming (G91) can be used in conjunction with circular interpolation. With G91 the end point and the centre point of the circular move are referenced to the start point programmed in the previous block.
The direction (CW or CCW) of a circular interpolation is defined by looking in the positive direction of the axis that is perpendicular to the active interpolation plane.The following examples show the directions for circular interpolation on the active planes.



Directions of a circular interpolation

Circular interpolation in absolute programming with the I and J coordinates of the centre of the circle.

```
N14	X10 Y20
N15	G2 X46 Y20 I28 J20 F200
N16	G3 X64 Y38 I46 J38
```



Circular interpolation in absolute programming with the value R of the radius of the circle.

```
N14	X10 Y20
N15	G2 X46 Y20 R18 F200
N16	G3 X64 Y38 R18
```

Circular interpolation in incremental programming with the coordinates J and I.

```
N14	X10 Y20
N15	G2 G91 X36 I18 J0 F200
N16	G3 X18 Y18 I0 J18
```

Circular interpolation in incremental programming with the value of the radius R.

```
N14	X10 Y20
N15	G2 G91 X36 R18 F200
N16	G3 X18 Y18 R18
```

CET (PRC) - Circular Endpoint Tolerance

In circular interpolations, CET defines the tolerance for the variance between the starting and final radiuses of the circle arc.

Syntax

CET=*value*

---

where:

value	Tolerance expressed in millimetres. The default value is 0.01 mm.

Characteristics:

If the difference between starting and final radius is smaller than the tolerance but not zero, the system normalises the circle data according to the values specified in CET and ARM.

If the difference is equal to or greater than the value assigned to CET, an error occurs and the programmed final points will not be executed. In this case, you must either modify the program or increase the CET tolerance.

The value assigned to CET can be modified as follows:

- In the AMP configuration
- By means of a specific data entry
- By writing a new CET in the part program.

The CET tolerance is always expressed in the characterised measuring unit (G70/G71 apply).

If the variance between programmed start and final radius is higher than the CET value, the circle arc can be executed as follows:

- By making the CET value greater than the actual variance
- By programming the arc with the circle radius rather than with the centre using this format:

  G2/G3, final point and R radius

A RESET re-establishes the default tolerance.

Example:

CET=0.02	defines a 0.02 mm tolerance

FCT - Full Circle Threshold

In a circular interpolation, the FCT instruction defines a threshold for the distance between the first and the last point in an arc. Within this distance the arc is considered a full circle.

Syntax

FCT=*value*

where:

value	Threshold expressed in millimetres. The default value is 0.001 mm.

Characteristics:

The FCT command allows to deal with inaccurate program data that would otherwise prevent the system from forcing a complete circle. In other words, if the distance from the first to the last point is less than FCT, the system uses the points as if they were overlapping and forces a full circle.

FCT thresholds can be modified as follows:

- In the AMP configuration
- By means of a specific data entry
- By writing a new CET in the part program.

The FCT threshold is always expressed in the characterised measuring unit (G70/G71 apply).

A RESET re-establishes the default threshold.

Example:

```
G71
FCT=0.005
```

In this example, FCT defines a threshold 0.005 millimetres.

ARM - Defining Arc Normalisation Mode

The ARM code defines the method with which the system normalises an arc (programmed with the centre coordinates I and J, and a final point) in order to render it geometrically congruent.

An arc is normalised when the variance between initial and final radius is less than the characterised accuracy tolerance or than the tolerance programmed with the CET command.

Before executing an arc, the system calculates the difference between initial and final radiuses.

- If the difference is zero, the control will execute the programmed arc without normalising it.
- If the difference is greater than the CET value, the control will stop without executing the move, and display a profile error message.
- If the difference is less than the CET value, the control will execute the move normalising the arc with the method specified by ARM.
- If the distance is less than the FCT threshold, the system will force the complete circle. For ISO blocks with radius compensation, the system checks the difference twice: first on the base profile without compensation (normalisation stage) and then on the compensated profile (motion generation stage).

Syntax

ARM=*arc mode*

---

where:

arc mode	Is the numerical value that defines the arc normalisation mode. Valid values are:

0	displaced centre within the CET tolerance (default mode)
1	displaced starting point displaced the CET tolerance
2	displaced centre independent from the CET tolerance
3	centre beyond the CET tolerance range

The default value is zero.

Characteristics:

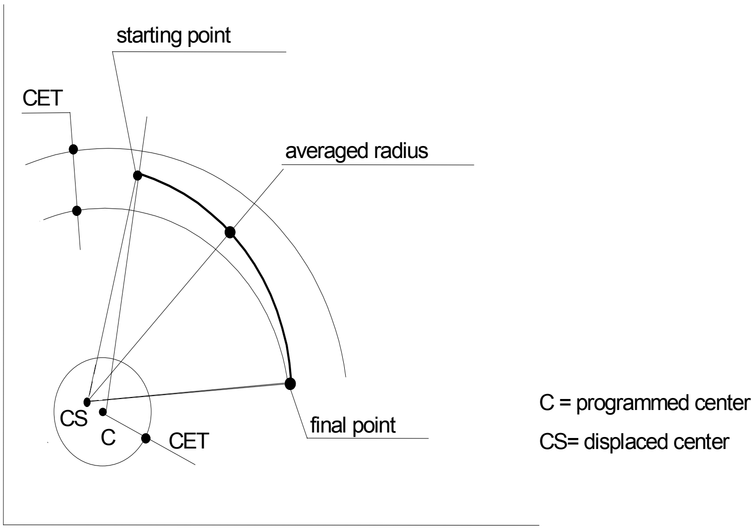
The arc normalisation mode can be modified as follows:

- In the AMP configuration
- By means of a specific data entry
- By writing a new CET in the part program.

The examples that follow illustrate ARC normalisation modes.

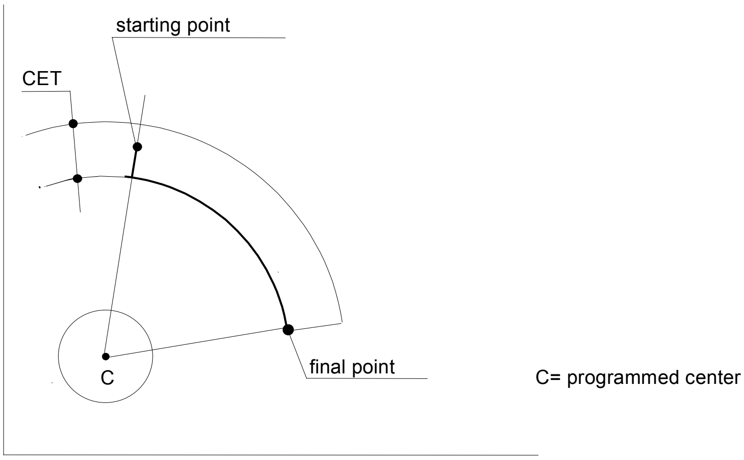
ARM=0

This is an arc through the initial and final programmed points whose centre is displaced within the tolerance defined by CET. The arc is executed with averaged radius.



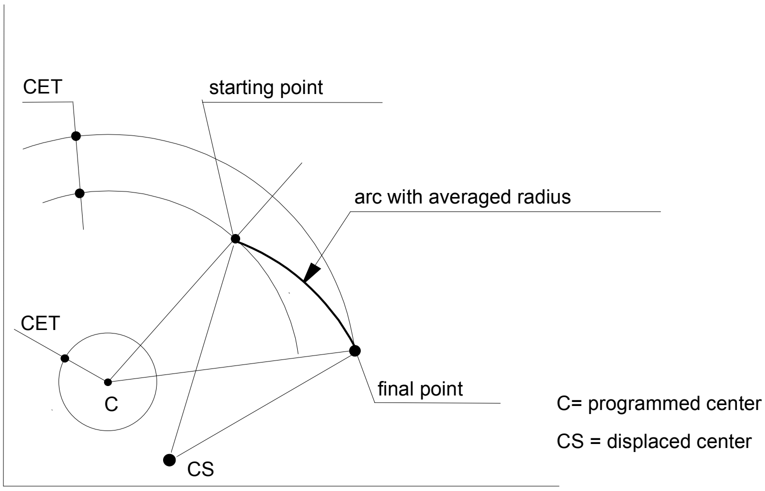
ARM=1

This is an arc through the programmed final point and the starting point displaced within the CET tolerance. The arc is executed with final radius.



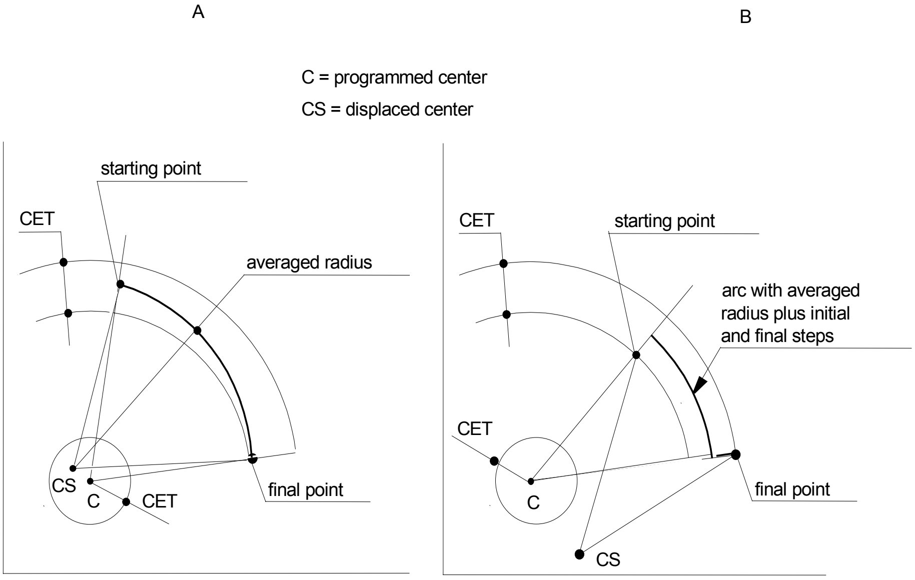
ARM=2

This is an arc whose centre is displaced irrespective of the tolerance defined with CET. In this case the arc is executed with averaged radius.



ARM=3

If the displacement of the centre arc is within the CET tolerance defined with CET, the arc centre will be displaced and the arc will pass through the programmed starting and final points. If the displacement of the centre is not within the CET tolerance, the arc will have the programmed centre and pass through the displaced starting and final points (both points are displaced within the CET/2 tolerance).
In this case the arc is executed with averaged radius.



IMPORTANT

With ARM = 1 or ARM = 3 the resultant profile can show inaccuracies ("steps"):

With ARM = 1 there will be a step at circle start equal to the difference between starting and final radiuses.

In case of ARM = 3 there will be a step both at circle arc start and end.

To prevent these steps from causing a servo error, we suggest that you program a CET value smaller than the characterised servo error threshold.

CRT - Circular interpolation speed reduction threshold
CRK - Circular interpolation speed reduction constant

The variables CRT (Circle Reduction Threshold) and CRK (Circle Reduction K-Constant) are used for reducing the speed on circular elements by applying different reduction algorithms depending on the sign of the CRT parameter. Speed reduction will be:

- as a function of the radius of the element only, if CRT > 0
- as a function of the radius and centrifugal acceleration, if CRT = 0
- as a function of the radius and tolerated interlocking error, if CRT < 0

Syntax

CRT = *value*

---

where:

value	If value > 0 => value, it is the threshold radius below which the reduction must be applied. A value of 0 (zero), which is the default value, cancels this operation. If value < 0 => abs(value), it is the maximum departure in mm desired between the programmed and the actual path. A value of 0 (zero), which is the default value, cancels this operation.

CRK = *value*

---

where:

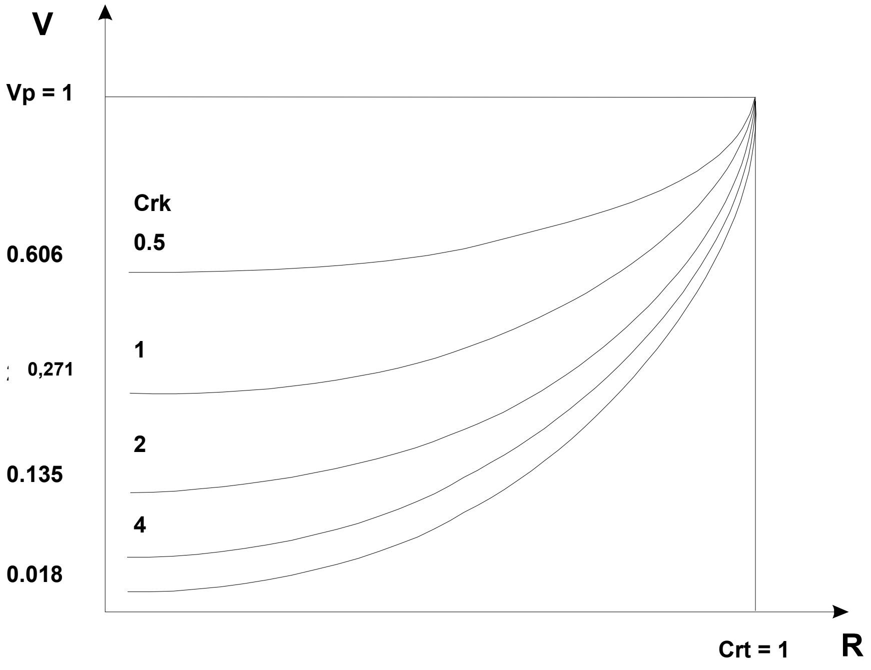
value	if CRT > 0 =>	is a constant for modulating the reduction in speed. The value set by default is 1.
	if CRT = 0 =>	Is a constant for modulating the reduction in speed depending on centrifugal acceleration.
	if CRT < 0 =>	Is the axis position interlocking gain (Kv). The value to be specified must be the same as configured in AMP in the corresponding axis field.

If both CRT and CRK are nil, then no reduction speed is applied to the circular elements.

Characteristics:

CRT is the variable identifying the type of strategy to be adopted to reduce speed on circles. If:

- **CRT > 0** => By assigning any value positive to the variable CRT, the speed is reduced on all circular elements with a smaller radius than the value set. The value assigned to the variable CRK enables this reduction to be modulated. The speed is reduced as shown in the graph below, in which it is assumed that the programmed speed Vp is equal to 1 and the variable CRT is equal to 1.



- **CRT = 0** => When CRT becomes 0, the CRK value (if other than zero) is used, in circular movements, in order to recalculate processing speed. In circular movements, in fact, processing speed is generally limited by the radius of the circumference (centrifugal acceleration) according to the following relationship:

  $$V\ lav = Min\ (\sqrt{a\ radius},\ V\ Prog\ )$$

  where a is the minimum acceleration between the two axes involved in the circular movement.

  CRK changes this relationship as follows

  $$V\ lav = Min\ (\ CRK * \sqrt{a\ radius},\ V\ Prog\ )$$

  and therefore makes it possible to increase (CRK > 1.0) or decrease (CRK < 1.0) the limitation associated with the centrifugal acceleration. With a value of 1.0 the standard calculation is retained.

- **CRT < 0** => By assigning any value negative to the variable CRT, the speed is reduced on all circular elements as a function of the curvature radius of the path in order to make sure the error does not exceed the limit specified in CRT.

  Hence, maximum admissible speed will be a function of the radius, the error specified by the absolute value of CRT and the interlocking gain CRK, according to the following formula:

  $$\mathbf{Vmax = CRK * \sqrt{2 * R * abs(CRT)} * f(VFF,FLT\_2)}$$

  Where f(VFF,FLT_2) is a function that depends on the Velocity Feed Forward (VFF) set and the type 2 filter, if any, activated (centripetal acceleration compensation filter). For further details, see the Chapter on filters in this manual.

  NOTE:

  Since speed on circles is limited as a function of the percentage of VFF, it is necessary, when working with VFF, to have enabled the process variable VFF and configured the same percentage value both on the drives and on the axes.

The values assigned to the variables CRT and CRK may be modified as follows

- by means of the AMP command during configuration
- from the part program with the specified syntax.

The values assigned to CRT are always expressed in the current unit of measurement of the process (the G70/G71 functions are applied).

The RESET command restores the characterization values.

Helical Interpolation

G02 and G03 program a helical path in only one block. The system performs the helical path by moving the plane axes in a circular interpolation while the axis that is perpendicular to the interpolation plane moves linearly.

To program a helical path, simply add a depth coordinate and the helix pitch (K) to the parameters specified in the circular interpolation block.

Syntax

G02 [*G-codes*] [*axes]* **I**.. **J**.. **K**.. [**F**..] [*auxiliary*]
or
G02 [*G-codes*] [*axes]* **R**.. **K**.. [**F**..] [*auxiliary*]

G03 [*G-codes*] [*axes*] **I**.. **J**.. **K**.. [**F**..] [*auxiliary*]
or
G03 [*G-codes*] [*axes*] **R**.. **K**.. [**F**..] [*auxiliary*]

where:

G-codes	Other G codes that are compatible with G02 and G03 (See "Compatible G codes" table in Chapter 1).
axes	An axis letter followed by a numerical value programmed (either decimal value or E parameter). If no axes are programmed in the block, the move will generate a full circle on the active interpolation plane.
I	Abscissa of the circle centre. This is a value in millimetres (decimal number or E parameter). The abscissa is expressed as a diameter unit when the corresponding axis is a diameter axis. No matter what the interpolation plane, the symbol for the abscissa is always **I**.
J	Ordinate of the circle centre. This is a value in millimetres (decimal number or E parameter). The ordinate is expressed as a diameter unit when the corresponding axis is a diameter axis. No matter what the interpolation plane, the symbol for the ordinate is always **J**.

R	Circle radius. It is specified with the R address followed by a length value, and is alternative to the I and J coordinates.
K	Helix pitch. This parameter is specified with the K address followed by the pitch value. It can be omitted if the helix depth is less than one pitch.
F	Feedrate. It is specified by the F address followed by a value. If it is omitted, the system will use the previously programmed feedrate. If no feedrate has been programmed, the system will signal an error.
auxiliary	Programmable M, S, and T functions. Up to four M functions, one S (spindle speed) and one T (tool selection) can be programmed in the block.

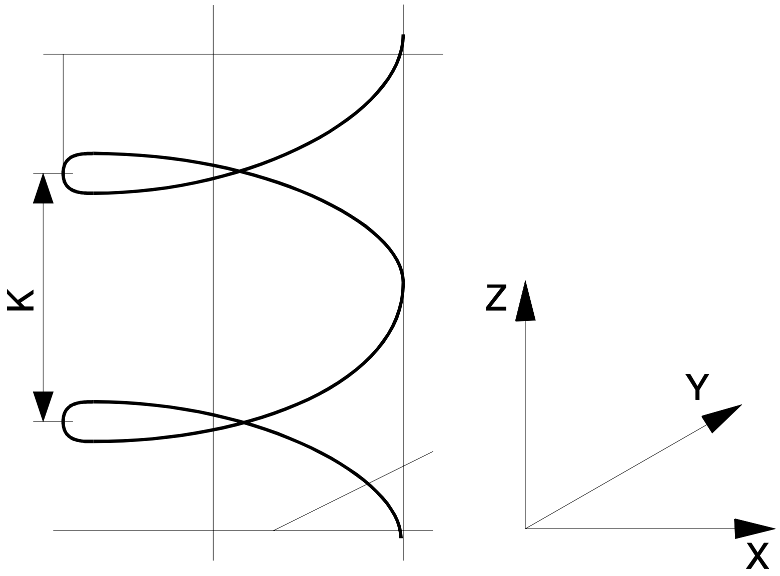
Characteristics:

If Z is a multiple of K, it is not necessary to program the final point

If the depth is not an integer number of pitches, i.e. if Z is not equal to n * K), the length of the circle arc must be calculated with the decimal remainder of the pitch number. For example, if Z = 2.7 * K, then the arc that must be programmed is 360 * (2.7 - 2) = 252 degrees.

Example:
G2 X . . Y. . Z . . I . . J . . K . . F. .

In this example, addresses X, Y, I, and J refer to circle programming; addresses Z and K refer to helix programming and are respectively the depth and the helix pitch. The figure below shows the typical dimensions of a helical interpolation.



Dimensions Helix

G33 - Constant or Variable Pitch Threading

G33 defines a cylindrical, taper, or face threading movement with constant or variable pitch. The threading move is synchronised to spindle rotation. The parameters programmed in the block identify the type of thread.

Syntax

G33 [*axes*] **K**.. [**I**..] [**R**..]

where:

axes	An axis letter followed by a numerical value.
K	Thread pitch (mandatory). For variable pitch threads, K is the initial pitch.
I	Pitch variation for variable pitch threading. For increasing pitch threading, I must be positive; for decreasing pitch threading **I** must be negative.
R	Deviation from the zero spindle angular position in degrees. R is used in multistart threading to avoid displacing the starting point.

Characteristics:

All these numerical values can be programmed directly with decimal numbers or indirectly with E parameters.

In decreasing pitch threads, the initial pitch, the pitch variation, and the thread length must be calculated so that the pitch is greater than zero before reaching the final coordinate. Use the following formula:

$$I \leq \frac{K^2}{2\,(Zf - Zi)}$$

where:

I	Is the maximum pitch variation
K	Is the initial pitch
(Zf - Zi)	Is the thread length.

IMPORTANT

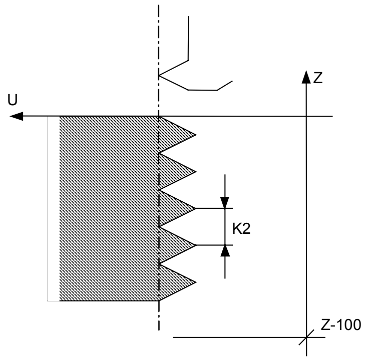
During the threading cycle the control ignores the CYCLE STOP button and the FEEDRATE OVERRIDE selector/softkey, whereas the SPINDLE SPEED OVERRIDE selector must be disabled by the machine logic.

VFF may be disabled with the dedicated softkey or with a VFF command.

Constant Pitch Threading

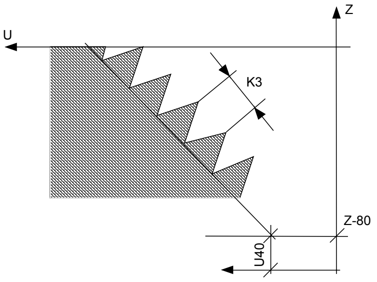
The figures that follow illustrate examples of constant pitch threading. Note that the U axis is a diameter axis.

Cylindrical threading



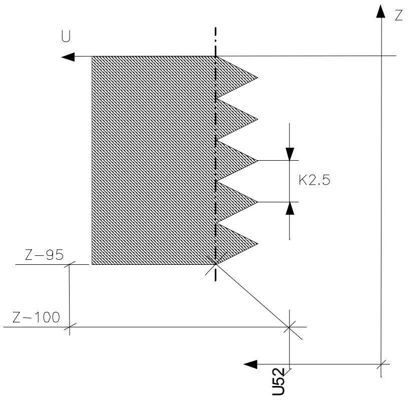
Part program block: G33 Z-100 K2

Conical threading



Part program block: G33 U40 Z-80 K3

Cylindrical-conical threading

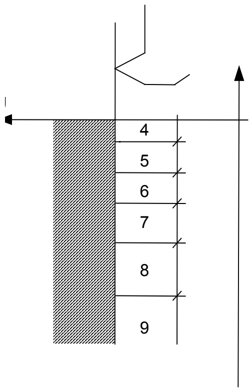


Part program blocks: G33 Z-95 K2 .5
Z-100 U52 K2.5

Variable Pitch Threading

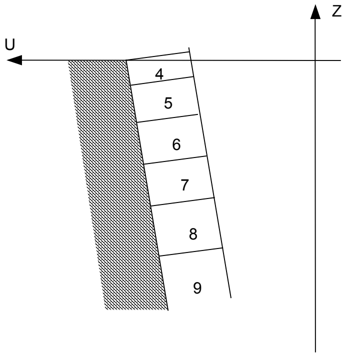
The figures that follow illustrate variable pitch threading. Note that the U axis is a diameter axis.

Cylindrical threading with increasing pitch



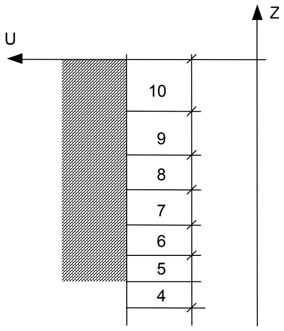
Part program block: G33 Z-50 K4 I1

Conical threading with increasing pitch



Part program block: G33 U50 Z-40 K4 I1

Cylindrical thread with decreasing pitch



Part program block: G33 Z-50 K10 I-1

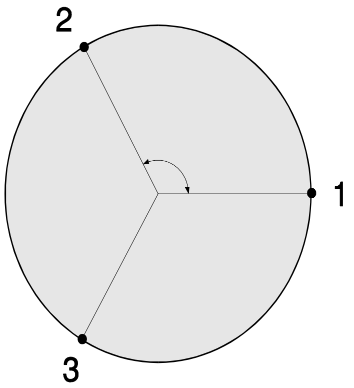
Multi-start threading

An R word in a G33 block makes the control start moving the axes from an angular position that varies according to the programmed R value.

This permits to program the same start point for all threads, rather than move the start point of each thread by a distance equal to the pitch divided by the number of starts.

Example:
Three-start threading

```
N37 G33 Z3 K6          1st thread
          .
          .
          .
N41 G33 Z3 K6 R120     2nd thread
          .
          .
          .
N45 G33 Z3 K6 R240     3rd thread
```



Rotary Axes

In the system characterisation, axes can be configured as rotary axes, i.e. a rotary table.

To program rotary axis moves simultaneous to and coordinated with the other axes programmed in the same block:

- Always use decimal degrees (from +0.00001 to +99999.99999 degrees) starting from a pre-selected origin.
- Select either the rapid rate (G00) or the feedrate (G01). In a rotary move rates are always expressed in degrees per minute (dpm, F5.5 format). For example, with F75.5 the axis moves at 75.5 dpm.

To perform milling operations on a circle with a rotary table, calculate the rotary rate with the following formula.

$$F = \frac{360}{pi} * \frac{A}{D} = 114{,}64 * \frac{A}{D}$$

where:

F	Is the rotary rate in dpm
A	Is the linear rate on the arc in millimetres or inches per minute
D	Is the diameter on which the milling operation is performed (in mm or inches).

To move rotary and linear axes simultaneously in the same block, you may calculate the feedrate with one of the following formulas.

With G94:

$$F = A * \frac{\sqrt{X^2+Y^2+Z^2+B^2+C^2}}{L}$$

where:

F	Is the feedrate
A	Is the feedrate on the part (in mm/min or inches/min)
X Y Z B C	Is the actual travel performed by each axis (in mm or inches for linear axes, in degrees for rotary axes)
L	Is the resultant path length (in mm or inches).

With G93:

$$F = \frac{A}{\sqrt{X^2 + Y^2 + B^2}}$$

where:

F	Is the feedrate
A	Is the desired feedrate (in mm/min or inches/min) on the part
X	Is the X axis incremental distance
Y	Is the Y axis incremental distance
B	Is the B axis incremental distance

The control cannot calculate the desired tool feedrate directly because the radius is not programmed. In these cases, the feedrate can be specified as inverse time with G93.

A block moving only the rotary axes generates an arc. If rotary and linear moves are combined, the resulting path may be an Archimedean spiral, a cylindrical helix or more complex curves, depending on the programmed number of linear axes.

Axes with Rollover

Axes with rollover axis are rotary or linear axes whose position is controlled between zero and a positive value configured in the *rollover pitch* parameter.

In the following description the axsi with rollover is rotary and has a 360 degree rollover pitch. We assume that the axis position is controlled in the 0 to 359.9999 degree range. That is, when the axis reaches 360 degrees, the displayed position rolls over to zero degrees.

An axis with rollover can be programmed in a block or in a MDI in two different modes:

absolute mode (G90)	programs the move in degrees.
incremental mode (G91)	programs the move as increments in degrees from the current axis position.

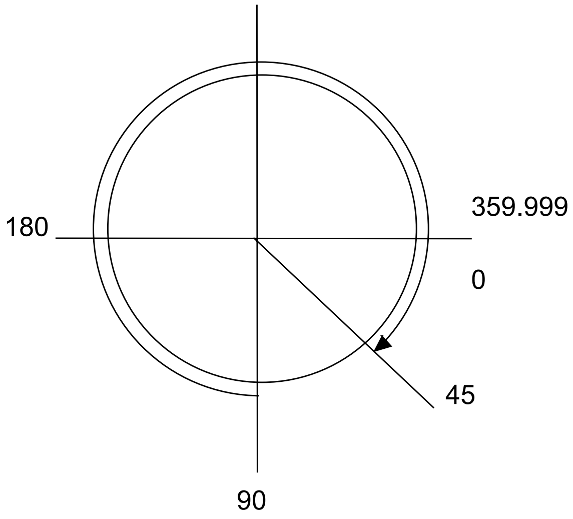
G90 - Absolute mode

In this mode:

- Displayed position is from 0 to +359.99999 degrees
- Programmed range is from 0 to ± 359.99999 degrees
- Direction of axis rotation depends on the sign of the programmed move. By convention, a positive move is clockwise and a negative move is counter clockwise.

For example, let's assume that rotary axis B is positioned at 90 degrees and the following block is written in the part program or in an MDI:

G90 B45

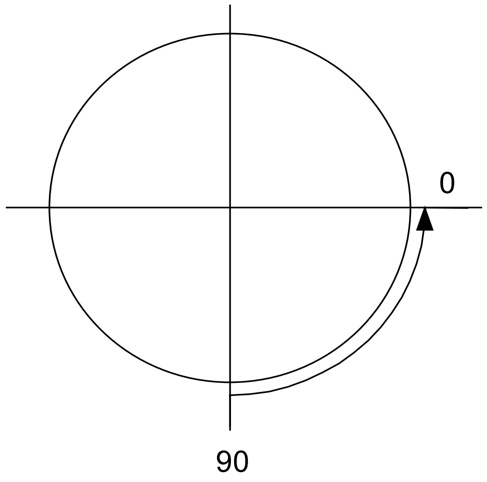


Clockwise Rotation

The B axis rotates by 315 degrees clockwise from the 90 degree position to reach the absolute position of 45 degrees (the sign of the move is positive).

Now let's assume that the B rotary axis is at 90 degrees and, the following block is written in the part program or in an MDI:

G90 B-0



Counter clockwise Rotation

The B axis rotates by 90 degrees counterclockwise to absolute position 0 degrees because the sign of the move is negative.

G91 - Incremental mode

When an axis with rollover is programmed in incremental (G91) mode, the following conditions apply:

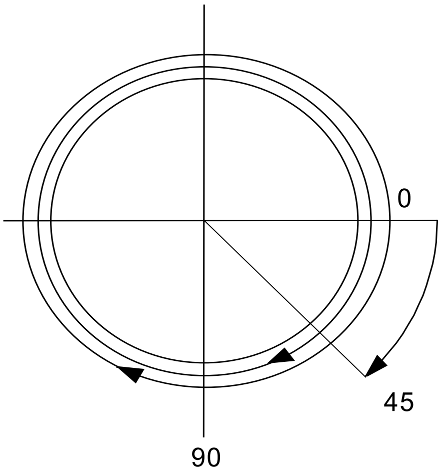
- Displayed position is from 0 to +359.99999 degrees.
- Program range is from +/-0.00001 to +/-99999.99999 degrees.
- Direction of axis rotation depends upon the sign of the programmed move. By convention, a positive move is clockwise and a negative move is counter clockwise.

IMPORTANT

The displayed position is beyond the programmed range when the programmed range is greater than +359.999 degrees.

For example, if the absolute zero position of the B rotary axis is 0 degrees and the following block is written in the part program or in an MDI:

G91 B765



Incremental clockwise rotation

The B axis makes two complete clockwise revolutions plus 45 degrees (360 + 360 + 45 = 765).

Pseudo Axes

A pseudo axis is an auxiliary function that may be addressed as an axis and is handled by the machine logic. The pseudo axis name can be any allowed axis name (X,Y,Z,A,B,C,U,V,W,P,Q,D). In a part program block it is possible to program up to 3 pseudo axes but in the AMP it is possible to configure up to 6 pseudo axes.

Diameter Axes

A reaming/facing head can be mounted on the spindle and controlled simultaneously with other axes. By programming such an axis (typically a U axis) as a diameter, the following can be obtained:

- Boring operations on cylindrical or conical holes
- Circular radiuses (concave or convex)
- Chamfers
- Grooves
- Facing operations
- Threads

Programming a U (diameter) axis is similar to programming other linear axes; however, its coordinates must be expressed in diameters. The measuring units can be inches or millimetres according to the current mode (G70/G71).

When the U axis is programmed in the same block as an X, Y or Z move, it is simultaneous with and coordinated to the other axes. U axis moves can be performed at rapid rate (G00) or feedrate (G01) with F in ipm or mmpm.

Before executing a profile with the U axis, the interpolation plane must be defined with the following command:

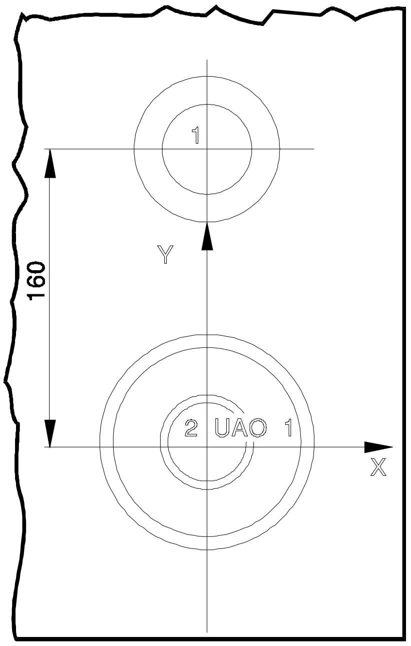
G16 Z U

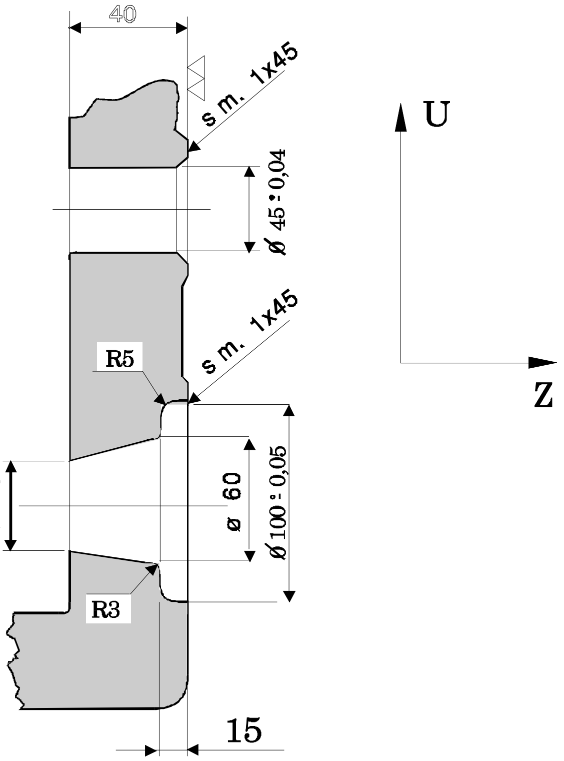
IMPORTANT

The order of Z and U in this command is critical, i.e. G16 UZ and G16 ZU define two different interpolation planes.

Cutter diameter compensation (G41 or G42) and a machining allowance (MSA) can be applied to profiles programmed with U.

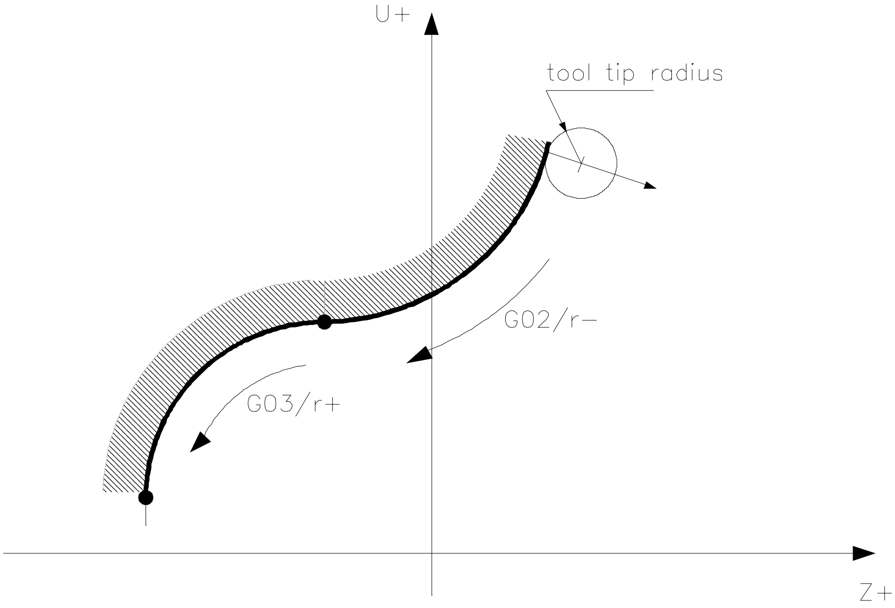
Example:
This is an example reaming/facing head used in a finishing operation.





```
N116 (DIS, "FINISHING WITH R/F HEAD")
N117 F60 S630 T9 .9 M6
N118 G16 Z U                ;Defines interpolation plane
N119 (UAO, 2)               ;Calls absolute origin for the head
N120 (UTO, 1, Z-200)        ;Temporary origin for Z (skimming the part)
N121 X Y160 M3              ;Position to hole 1
N122 G41 Z2 U51
N123 G1 Z-1 U44 .98         ;Executes the chamfer
N124 Z-44                   ;Executes hole diameter 45
N125 G G40 U40
N126 Z2 F40 S380
N127 G41 Y U106             ;Positions to hole 2
N128 G1 Z-1 U99 .975        ;Executes the chamfer
N129 Z-15                   ;Executes hole diameter 100
N130 r5                     ;Executes radius R = 5
N131 U60                    ;Executes counter boring
N132 r-3                    ;Executes radius R = 3
N133 Z-40 U40               ;Executes taper
N134 G40 Z-44               ;Continues Z axis travel
N135 G U35
N136 Z100 M5
N137 G16 X Y
N138 (UAO,1)
```

The direction of the arcs programmed with G02/G03 or with the r address and the direction for cutter diameter compensation (G41/G42) can be determined by looking at the profile on the Z-U plane. Since negative diameters are usually not programmed, you must consider only the first two quadrants of the plane.



UDA - Dual Axes

It is possible to treat one or more axes as slaves or subordinate to another defined as the master. In this way only the movements of the Master need be programmed as the movement of the slaves is determined by those of the Master to which they are associated and by whether or not reverse mirror movement has been applied.

Syntax

(UDA,*master1/slave1*[,*master2/slave2***,***master3/slave3***,***master4/slave4*]**)**
(UDA)

where:

master1. . . master4	Are the master axis names (one ASCII character per axis). You can program up to 4 master axes.
slave1. . . slave8	Are the slave axes names (one ASCII character per axis). You can program a maximum of 8 slave axes per master.
no parameters	(UDA) without parameters disables the dual axes mode (UDA)

Characteristics:

Dual axes management does not require any special setting in the system with AMP.
After an (UDA...) command, the positive operating limit is the minimum between the positive limit of the master axis and the current position of the master plus the distance that may be covered by the slave axis. In short:

PositiveLim = min(Master PositiveLim, MasterPosition + Slave PositiveLim - SlavePosition)

In the case of a "mirror", the distance that may be covered by the slave refers to its negative limit, so it will be:

PositiveLim = min(Master PositiveLim, MasterPosition - Slave NegativeLim + SlavePosition)

The considerations made for the positive limit also apply to the negative limit:

NegativeLim = max(Master NegativeLim, MasterPosition + Slave NegativeLim - SlavePosition)

In the case of a mirror, it will be:

NegativeLim = max(Master NegativeLim, MasterPosition - Slave PositiveLim + SlavePosition)

When the (UDA...) command is executed, reference must be made both to the master axis and the slave axes.

The RESET command does not cancel the association between the master and slave axes.

Dual axes may be used on rotated planes or in polar or cylindrical coordinates (UVP, UVC).

Dual axes, whether master or slave, must be defined on real axes and not virtual axes.

NOTE:
The names of masters and slaves must be separated by a / (slash).

IMPORTANT

To mirror the slave axis movement, you must program the - operator before the axis name. This rule does not apply to master axes.

Example:

(UDA,X/-U)	U is slaved and mirrored to X
(UDA, A/B - CD)	B, C, D are slaved to A and C is mirrored to A

SDA - Special Dual Axes

It is possible to treat one or more axes as slaves or subordinate to another defined as the master. In this way only the movements of the Master need be programmed as the movement of the slaves is determined by those of the Master to which they are associated and by whether or not reverse mirror movement has been applied. The movement of the master and slave axes can occur even if the axes are not referenced.

Syntax

(SDA,*master1/slave1*[,*master2/slave2***,***master3/slave3***,***master4/slave4*]**)**
(SDA)

---

where:

master1. . . master4	Are the master axis names (one ASCII character per axis). You can program up to 4 master axes.
slave1. . . slave8	Are the slave axes names (one ASCII character per axis). You can programa maximum of 8 slave axes per master.
no parameters	(SDA) without parameters disables the special dual axes mode (SDA)

Characteristics:

After an (SDA...) command, the positive operating limit is the minimum between the positive limit of the master axis and the current position of the master plus the distance that may be covered by the slave axis. In short:

PositiveLim = min(Master PositiveLim, MasterPosition + Slave PositiveLim - SlavePosition)

In the case of a "mirror", the distance that may be covered by the slave refers to its negative limit, so it will be:

PositiveLim = min(Master PositiveLim, MasterPosition - Slave NegativeLim + SlavePosition)

The considerations made for the positive limit also apply to the negative limit:

NegativeLim = max(Master NegativeLim, MasterPosition + Slave NegativeLim - SlavePosition)

In the case of a mirror, it will be:

NegativeLim = max(Master NegativeLim, MasterPosition - Slave PositiveLim + SlavePosition)

Upon activating the (SDA,...) command the master and the slaves need not be referenced.
The RESET command does not remove the master/slave association.

This allows one to perform the zero point micro-search cycle with dual movement active. In this case performing the zero point micro-search cycle, the system simultaneously moves the associated slaves. At the end of the search the master axis is referenced where as the slave axes are not. In order to reference the slave axes they should be exchanged, one by one, with the master axis by new SDA programming and the zero point micro-search cycle repeated for each one of them, redefined as the master. The initialisation of the slaves is not however necessary if one intends to program only the master with SDA active.

The use of the SDA function is recommended in cases where a zero point micro-search cycle is to be performed following a shutdown of the system with the work still on the work-bench.

NOTE:

The names of masters and slaves must be separated by a / (slash).

IMPORTANT

To mirror the slave axis movement, you must program the - operator before the axis name. This rule does not apply to master axes.

Example:

(SDA,X/-U)	U is slaved and mirrored to X
(SDA, A/B - CD)	B, C, D are slaved to A and C is mirrored to A

XDA - Master/Slave axes

One or more axes can be used as “slave” axes, i.e. subordinate to another axis, defined as “master”. In this manner, you only need to program the movements of the master, since the movements of the slaves are determined by the master they are associated with and by a following factor, which is defined specifically for each individual slave. Commands of different types can be imparted with this feature, each of them having a specific syntax.

Master/Slave Association

This instruction defines the association between a master axis and its slaves (up to 8 axes). It **does NOT activate the following function**, whose activation is by means of a specific command. This means that after this instruction a movement of the master does not bring about a movement of the slave(s). **After this instruction and until the slave is released from the master, slave movements cannot be programmed.**

Syntax

(XDA, 1, master/slave1[slave2[..]], mode, ratio, space)

---

where:

master	Is the name of the master axis and is denoted by a single ASCII character.
slave1…slave8	Are the names of the slave axes (each of them denoted by a single ASCII character). You can program up to 8 slaves.
mode	Defines the master axis following mode used by the slave(s). It can be: **0** The slave follows the master point by point **1** The slave follows the master in terms of speed **2** The slave follows the master in terms of position **3** The slave follows the master in terms of position, and the synchronisation distance is taken up
ratio	This is the master following ratio specified for the slave(s). It must be viewed as a multiplication factor for the feedrate of the master or the distance covered by it. If the value of this ratio is 1.0, the motion of the master is reproduced exactly by the slave; if it is smaller than 1.0, feedrate/distance are reduced, if it is greater than 1.0 they are increased. This value can be preceded by a sign.
distance	This is the distance to be covered by the slave to synchronise with the motion of the master.

Characteristics:

The master axis can identify either an axis present in the process in which the XDA command is activated or an axis which is not present; in the latter case, a "virtual" axis, having the name specified, will be created; this axis will have the dynamic characteristics inherited from the slave axes (the lowest values of feedrate, accelerations and jerk). This axis may be part of a virtualisation (UPR, UDA,…) and also of a TCP. The operation of master axis Homing cannot be performed.

At least one slave axis must be present in the process in which the XDA command is activated; it can be a SHARED axis, i.e. an axis shared with the machine logic environment. In this connection, the axis may continue to be moved by the machine logic even after the association with the master, however it cannot be moved while it is following the master. It cannot be part of any virtualisation or TCP.

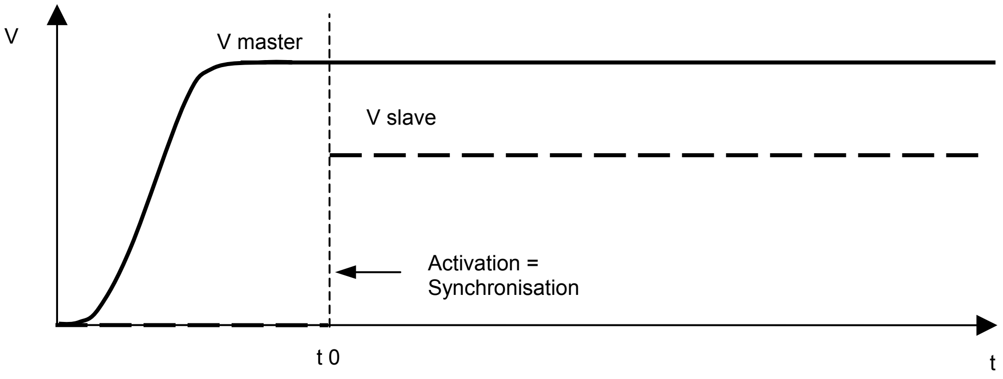
Let us now examine the various following modalities available:

Mode 0

In this mode, the slave axis follows the master proportionately to the value of the ratio (if the ratio = 1, the slave reproduces the movement of the master axis exactly), synchronisation is instantaneous and the variation in the feedrate of the slave is "*in steps*". Slave position and feedrate values are calculated, instant by instant, according to the following formulas:

$$\mathbf{Vslave = Vmaster * FollowRate}$$

$$\mathbf{PosSlave = PosSlave_{t0} + (PosMaster - PosMaster_{t0}) * FollowRate}$$

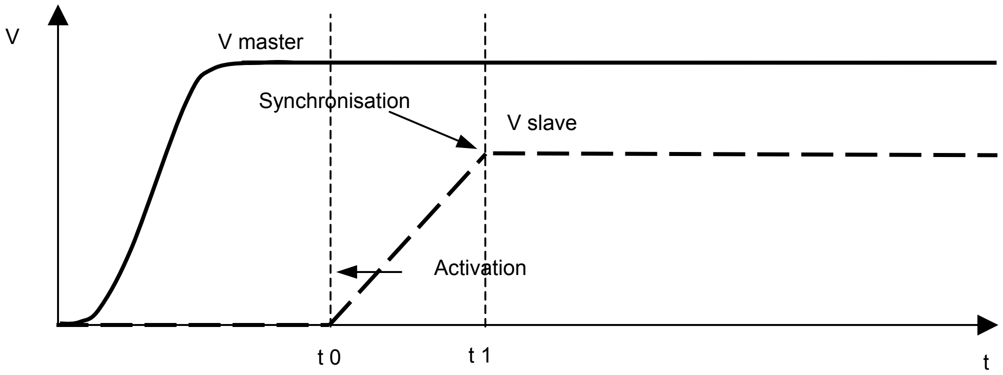


If the speed specified for the slave axis as a result of the following command exceeds the maximum admissible value for this axis, the system will reduce the feedrate requested accordingly and will give out an emergency (servo error) message, in that the slave is unable to follow the required position.

Mode 1

In this mode, the slave follows the feedrate of the master proportionately to the value of the ratio (if the ratio = 1, the slave copies the movement of the master axis exactly); the synchronisation depends on the dynamic characteristics of the slave axis and/or the distance parameter which defines the synchronisation distance.

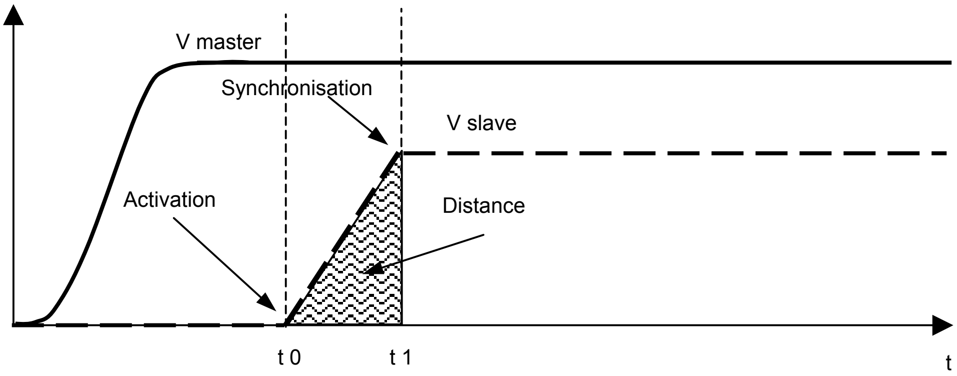
If the distance value = 0, the slave will synchronise with the master based on its maximum acceleration and using linear ramps **only**.



If the value is not 0, the slave will synchronise with the master based on an acceleration calculated as a function of the synchronisation distance and using linear ramps **only**. The acceleration will be calculated again with each sampling process based on the following formula

$$\mathbf{Aslave = ((Vmaster * FollowRate)^2 + Vslave^2) / 2 * Distance}$$

where the value of the distance is gradually reduced based on the distance covered during the synchronisation stage. No check is made on the ensuing acceleration value, and therefore servo errors may arise if the acceleration exceeds the maximum value that can be withstood by the axis.



Once the synchronisation with the master has taken place, the slave will move according to this formula:

$$\mathbf{Vslave = Vmaster * FollowRate}$$

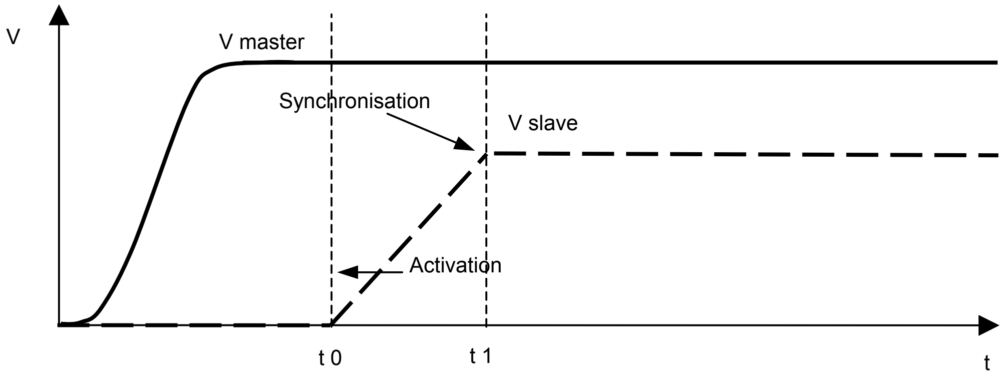
The feedrate (**Vslave**) determined in this manner is "theoretical", since it is necessary to determine whether this request is compatible with the dynamic characteristics of the axis (maximum feedrate and maximum acceleration). The moment the feedrate of the master varies, the slave will follow this variation based on its acceleration value. If the feedrate requested of the slave exceeds its maximum admissible feedrate, the system will reduce the feedrate requested accordingly. Hence, the feedrate and acceleration values with which the slave has to be moved, $\mathbf{Vslave}_i$ and $\mathbf{Aslave}_i$, will be determined instant by instant. The position of the slave will therefore be calculated on the basis of these values:

$$\mathbf{PosSlave}_{tn+1} = \mathbf{PosSlave}_{tn} + \mathbf{Vslave}_i + \mathbf{Aslave}_i$$

Mode 2

In this mode, the slave follows the position of the master proportionately to the value of the ratio (if the ratio = 1 the slave reproduces exactly the movement of the master); synchronisation depends on the dynamic characteristics of the slave axis and/or the distance parameter which defines the synchronisation distance.

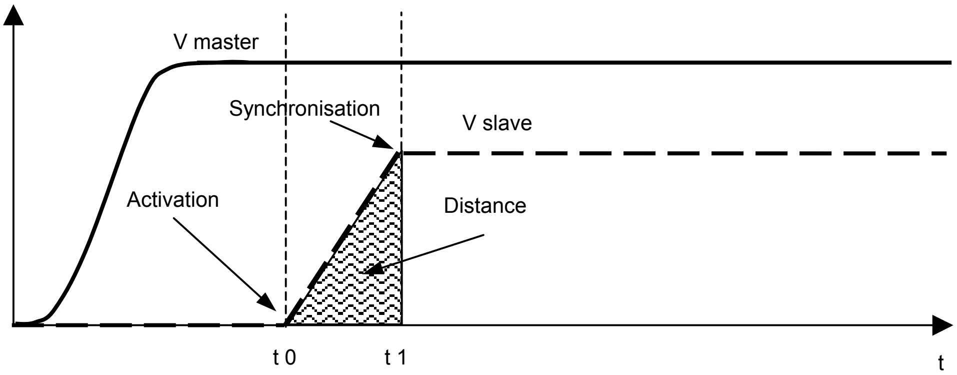
If the distance value is 0, the slave will synchronise with the master based on its maximum acceleration and using linear ramps **only**.



If the value of distance is not 0, the slave axis will synchronise with the master axis based on an acceleration calculated as a function of the synchronisation distance and using linear ramps **only**. The acceleration value will be calculated again with each sampling step according to this formula

$$\mathbf{Aslave = ((Vmaster * FollowRate)^2 + Vslave^2) / 2 * Distance}$$

where the value of the distance is gradually reduced based on the distance covered during the synchronisation stage. No check is made on the ensuing acceleration value and therefore servo error messages may be generated the moment the acceleration exceeds the maximum value that the axis can withstand.



Once the synchronisation with the master axis has occurred, the slave will move according to the following formulas:

$$\mathbf{PosSlave = PosSlave_{t1} + (PosMaster - PosMaster_{t1}) * FollowRate}$$

$$\mathbf{Vslave = Vmaster * FollowRate}$$

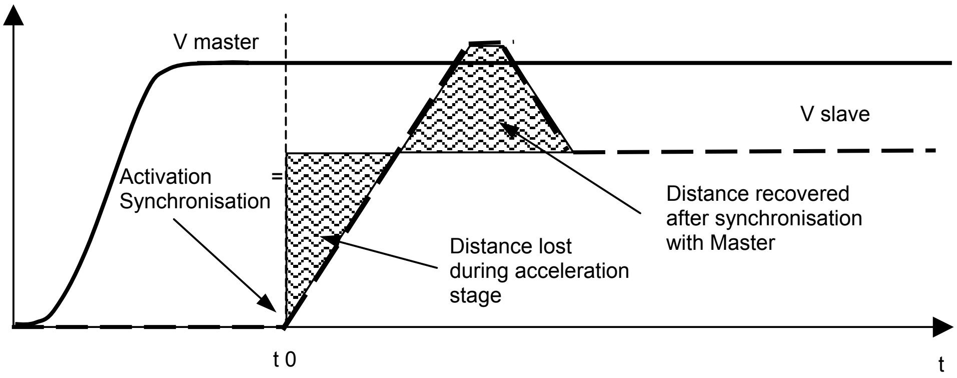
The position, **PosSlave**, and the feedrate, **Vslave**, determined in this manner should be rated as "theoretical" values, since it is necessary to determine whether the values requested are compatible with the dynamic characteristics of the axis (Maximum feedrate and maximum acceleration). The moment the feedrate of the master varies, the slave will follow this variation according to its own acceleration value. If the feedrate requested for the slave exceeds the maximum value admissible for this axis, the system will reduce the feedrate accordingly. To this end, the two values with which to move the slave, $\mathbf{Vslave_i}$ and $\mathbf{Aslave_i}$ will be calculated instant by instant. The actual position of the slave axis will therefore be calculated on the basis of these values:

$$\mathbf{PosSlave_{tn+1} = PosSlave_{tn} + Vslave_i + Aslave_i}$$

The difference between the actual and the theoretical position of the axis is taken up by the slave during its motion (even when the master has stopped moving) by moving, to the extent feasible, at a rate higher than the theoretical value (**Vslave**).

Mode 3

In this mode, the slave follows the position and feedrate of the master proportionately to the value of the ratio (if the ratio = 1, the slave reproduces exactly the movement of the master); synchronisation depends on the dynamic characteristics of the slave.



During the entire movement of the slave (i.e. both during and after the synchronisation stage), the motion of the axis is according to the following formulas (always using linear ramps):

$$\mathbf{PosSlave} = \mathbf{PosSlave_{to}} + (\mathbf{PosMaster} - \mathbf{PosMaster_{t0}}) * \mathbf{FollowRate}$$

$$\mathbf{Vslave} = \mathbf{Vmaster} * \mathbf{FollowRate}$$

The position (**PosSlave**) and the feedrate (**Vslave**) determined in this manner should be rated as "theoretical" values, in that it is necessary to determine whether these requests are compatible with the dynamic characteristics of the axis (max admissible feedrate and max admissible acceleration). The moment the feedrate of the master varies, the slave follows the variation according to its own acceleration value. If the feedrate requested of the slave is higher than its maximum admissible feedrate, the system reduces the feedrate requested accordingly. To this end, the two values with which the axis is to be moved ($\mathbf{Vslave}_i$ and $\mathbf{Aslave}_i$) will be calculated instant by instant. The actual position of the slave will therefore be calculated on the basis of these values:

$$\mathbf{PosSlave_{tn+1}} = \mathbf{PosSlave_{tn}} + \mathbf{Vslave_i} + \mathbf{Aslave_i}$$

The difference between the actual and the theoretical position of the axis is taken up by the slave during its motion (even when the master has stopped moving) by moving, to the extent feasible, at a rate higher than the theoretical value (**Vslave**).

Releasing the Slave(s) from the Master

This instruction removes the association between the master and the slave(s). **Following this instruction it will be possible to program any movement of the slave axis.**

Syntax

(XDA)

Defining/Changing the following ratio

This instruction defines/changes the parameter that determines the ratio according to which the master is followed by the slave(s) concerned.

Syntax

(XDA, 2, slave1[slave2[..]], ratio**)**
(xda, 2, slave1[slave2[..]], ratio**)**

---

where:

slave1…slave8	are the names of the slave axes (each of which is denoted by a single ASCII character). You can program up to 8 slaves.
ratio	This is the master following ratio specified for the slave(s). It must be viewed as a multiplication factor for the feedrate of the master or the distance covered by it. If the value of this ratio is 1.0, the motion of the master is reproduced exactly by the slave; if it is smaller than 1.0, feedrate/distance are reduced, if it is greater than 1.0 they are increased. This value can be preceded by a sign.

Characteristics:

The command can be used both when a slave is already following the master axis (it then brings about the release of the slave from the master and activates a new synchronisation stage using the new following parameter) and when the following function is not active (the command activates the following value to be used in the next movement stage).

If the uppercase syntax is used, the movement is stopped and the continuous command underway, if any, is terminated. If the lowercase syntax is used, instead, a continuous mode command is given out; at any rate, the axes are stopped at zero speed and after that are restarted immediately. If you do not want the movement to stop, this can be accomplished by having the machine logic execute a similar command.

Activating the following function

The following of the Master axis by the slave(s) is immediately activated. The following modality is defined by the “mode” parameter contained in the master/slave association command.

Syntax

(XDA, 3, slave1[slave2[..]]**)**
(xda, 3, slave1[slave2[..]]**)**

---

where:

slave1…slave8	are the names of the slave axes (each of which is denoted by a single ASCII character). You can program up to 8 slaves.

Characteristics:

If the uppercase syntax is used, the movement is stopped and the continuous command underway, if any, is terminated. If the lowercase syntax is used, instead, a continuous mode command is given out; at any rate, the axes are stopped at zero speed and are restarted immediately. If you do not want the movement to stop, this can be accomplished by having the machine logic execute a similar command.

Deactivating the following function

The following of the Master axis by the slave(s) is immediately deactivated. The release modality is defined by the “mode” parameter contained in the master/slave association command.

Syntax

(XDA, 4, slave1[slave2[..]]**)**
(xda, 4, slave1[slave2[..]]**)**

where:

slave1…slave8	are the names of the slave axes (each of which is denoted by a single ASCII character). You can program up to 8 slaves.

Characteristics:

The slave axis remains associated with the master, it just does not follow it any longer. Depending on the “*mode*” parameter defined in the master/slave association command, either of the following will occur:

0	The slave changes abruptly from the current feedrate to zero.
others	The slave comes to a halt according to its deceleration ramp.

If uppercase syntax is used, the movement is stopped and the continuous command underway, if any, is terminated. If lowercase syntax is used, instead, a continuous mode command is given out; at any rate, the axes are stopped at zero speed and are then restarted immediately. If you do not want the movement to stop, this can be accomplished by having the machine logic execute a similar command.

Example:

N10 (XDA,1,X/ZA,3,0.8,0.0)	Activates master X and slaves Z and A
N20 (XDA,3,ZA)	Activates following function by A and Z
N30 G1X100F2000	
N40 X300	
N50 (xda,4,Z)	Deactivates following function by Z in continuous mode
N60 X400	
N70 X500	
N80 (xda,4,A)	Deactivates following function by A in continuous mode
N90 X660	
N100 X700	
N110 (xda,3,ZA)	Reactivates following function by A and Z in continuous mode
N120 GX0	
N130 (XDA)	Removes association of slaves Z and A with master X
N140 GX	

AXF – Definition of axes with dynamic following function

With this command it is possible to define the axes to be managed separately from the others from the dynamic standpoint. The axes defined in the following triliteral describe the programmed geometry, but move independently of the other process axes.

Syntax

(AXF, *axle names***)**

(AXF)

where:

axle names Is a nominal list of the axes to which you want the following algorithm to be applied.

The triliteral without parameters disables the algorithm on all process axes.

Characteristics

The axes to which the dynamic following algorithm is applied are interpolated separately from the others, since with this triliteral two different interpolators are created: one for normal axes and one for the axes that follow.

The effect obtained is to prevent speed on the profile dropping to zero at the points where such axes start or end their movement, thereby making the entire process smoother, as shown in the example.

The axes that follow still have to be programmed.

This command is activated only if there is at least one axis to which the algorithm is not applied.

Each time the triliteral is programmed, any earlier following axis configuration is disabled.

WARNING

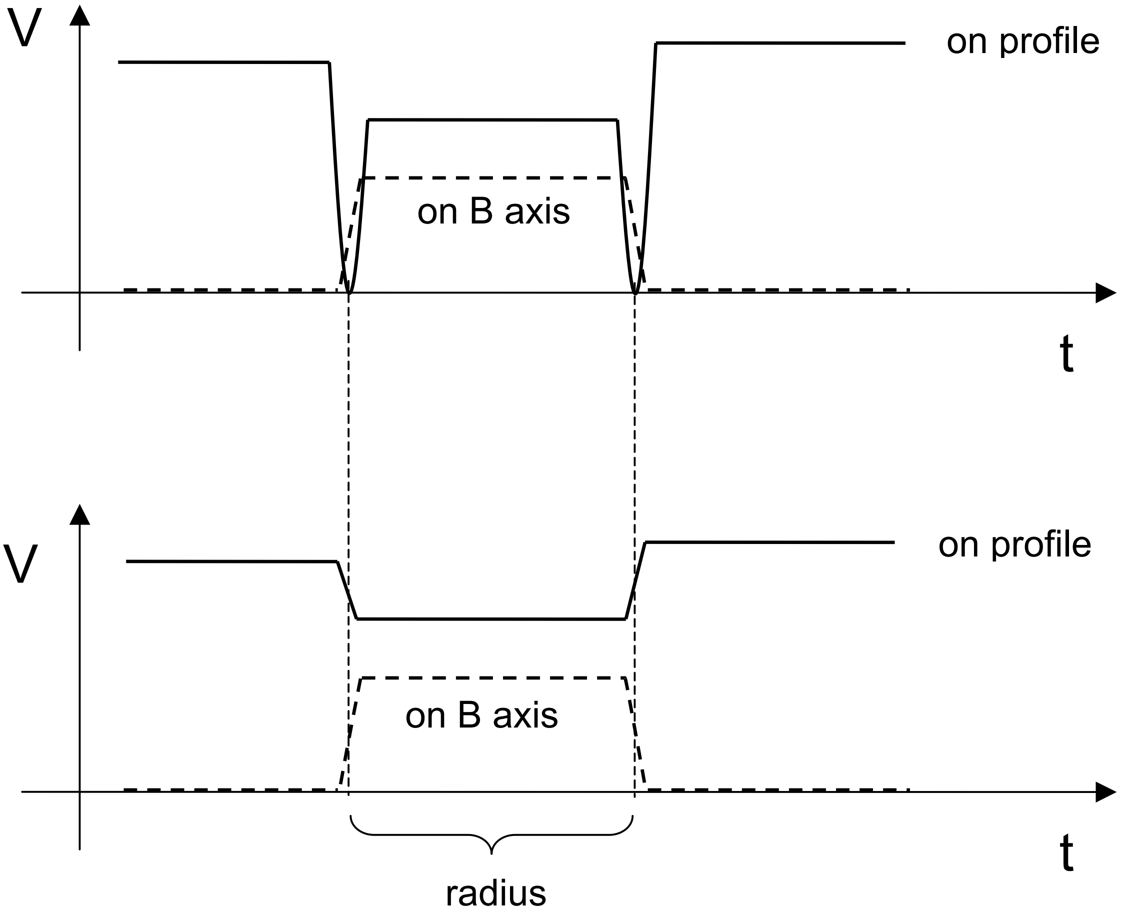
Rotary axes move according to the dynamic parameters configured for them. If the speed programmed for the profile exceeds maximum admissible speed, the axes cannot work properly (Servo Error). In this case, reduce the set speed.

Example:

Given a process with 3 linear axes (XYZ) and 1 rotary axis (B), let us assume that the process is programmed as described below:

(AXF,B)	After this command, the B axis is enabled to follow dynamically all the other axes of the process
G16XY G1 X100 YZAB F5000 G3 X150 Y50 I100 J50 B90 G1 Y150	A rounded corner is programmed, which becomes part of the movement of the rotary axis: • **without** the **AXF** command, the linear axes, at the start and end of the radius, come to a halt, as axis B is added in the interpolation • **with** the **AXF** command, the movement of B starts and ends on the radius but does not limit the dynamics of the linear axes: the speed on the profile does not drop to 0
(AXF)	After this command, rotary axis B is interpolated again together with the other process axes

The chart shows the evolution of speed on the profile and the B axis, respectively, for the two cases described above.



ORIGINS AND COORDINATE CONTROL CODES

The functions in this class perform the following operations:

G CODE	FUNCTION
G04	Dwell at end of step
G09	Deceleration at end of step
G16	Define interpolation plane
G17	Circular interpolation and cutter diameter compensation on XY plane
G18	Circular interpolation and cutter diameter compensation on ZX plane
G19	Circular interpolation and cutter diameter compensation on YZ plane
G27	Continuous movement with automatic velocity reduction on bevel
G28	Continuous movement without automatic velocity reduction on bevel
G29	Point-to-point movements
G70	Programming in inches
G71	Programming in millimetres
G79	Programming referred to axes home switch
G90	Absolute programming
G91	Incremental programming
G92	Axis presetting
G93	Inverse time (V/D) feedrate programming mode
G94	Feedrate programming in ipm or mmpm
G95	Feedrate programming in ipr or mmpr

NOTE:
The planes specified in G17, G18, G19 are valid if they have been configured in the following sequence: X, Y and Z.

G17 G18 G19 - Selecting the Interpolation Plane

These G codes are used for defining the interpolation plane as described below:

G17	Active interpolation plane formed by axes 1 and 2 (XY).
G18	Active interpolation plane formed by axes 3 and 1 (ZX).
G19	Active interpolation plane formed by axes 2 and 3 (YZ).

Axes 1 (X), 2 (Y), and 3 (Z) are the first three axes declared in the AMP environment.

Syntax

```
G17
G18
G19
```

The syntax for each function is simply the G code by itself in one block without parameters or other pieces of information.

G16 - Defining the Interpolation Plane

Like G17, G18, and G19, G16 defines the abscissa and the ordinate of the interpolation plane but is not linked to the first and second configured axes.

Syntax

G16 *axis1 axis2*

---

where:

axis1	Is the name of the abscissa of the interpolation plane (typically X). It must be one of the configured axes in the system.
axis2	Is the name of the ordinate of the interpolation plane (typically Y). It must be one of the configured axes in the system.

Characteristics:

G16, G17, G18, G19 cannot be used if the following G codes are active:

- Cutter diameter compensation (G41-G42)
- Standard canned cycles (G81-G89)

Example:

G16 X A	specifies the interpolation plane formed by axes X and A .

G27 G28 G29 - Defining the Dynamic Mode

The G functions in this class define how the axis moves on the profile and positions at profile end. These codes are always accepted by the control.

G27	Specifies a continuous move with automatic velocity reduction on bevels. At the end of each element velocity is automatically calculated by the control and optimised according to the profile shape. This calculation is based on DLA, MDA and VEF values.
G28	Specifies a continuous move without automatic velocity reduction on bevels. At the end of each element the velocity on the profile is equal to the programmed feedrate.
G29	Specifies a point-to-point move that is independent from the programmed path function (G01-G02-G03). At the end of each element the velocity on the profile is 0.

Syntax

G27 [*G-codes*] [*operands*]

G28 [*G-codes*] [*operands*]

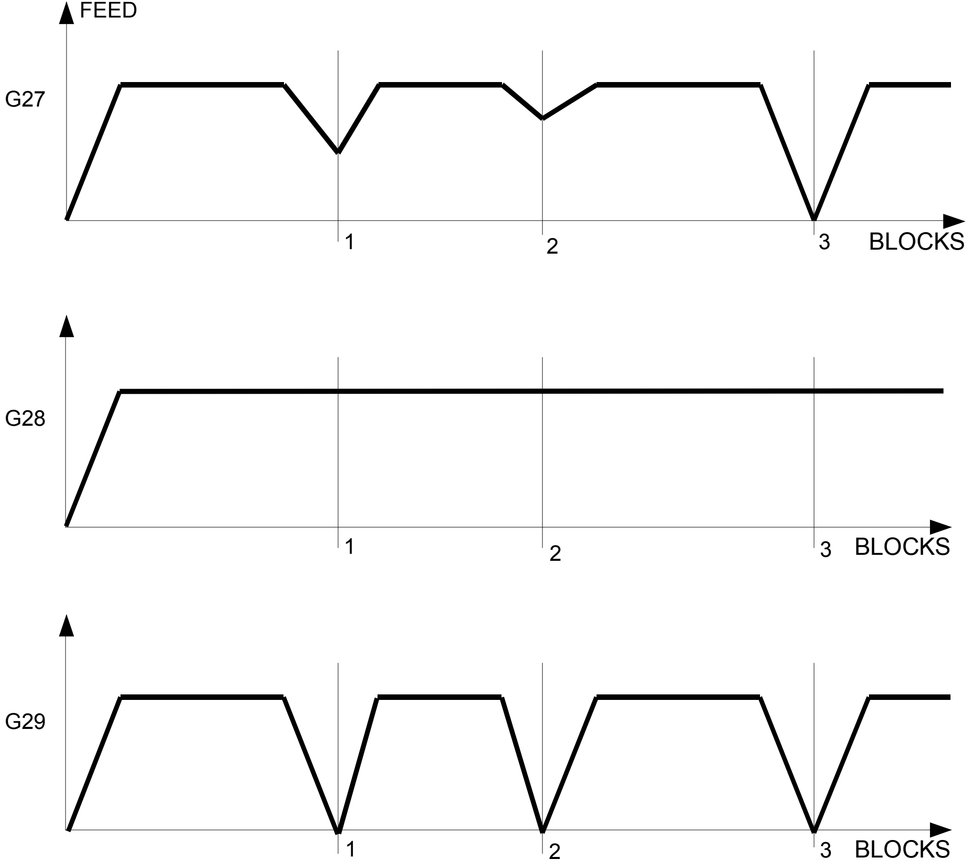
G29 [*G-codes*] [*operands*]

where:

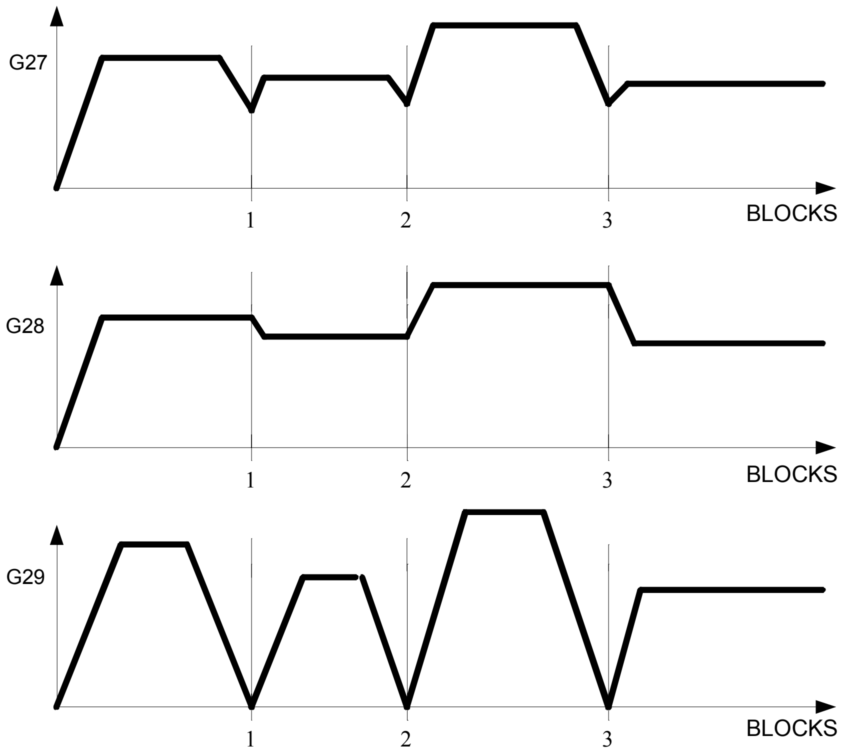
G-codes	Other G codes that are compatible with G27, G28 and G29 (See "Compatible G codes" table in Chapter 1).
operands	Any operand or code that can be used in a G function block.

Characteristics:

The following diagram shows how G27, G28 and G29 operate when the programmed feedrate is constant throughout the profile.



The following diagram shows how G27, G28 and G29 operate when the programmed feedrate varies through the profile.



In each block the move is divided into three steps:

1) Acceleration

2) Uniform move at programmed feedrate

3) Decelerated motion

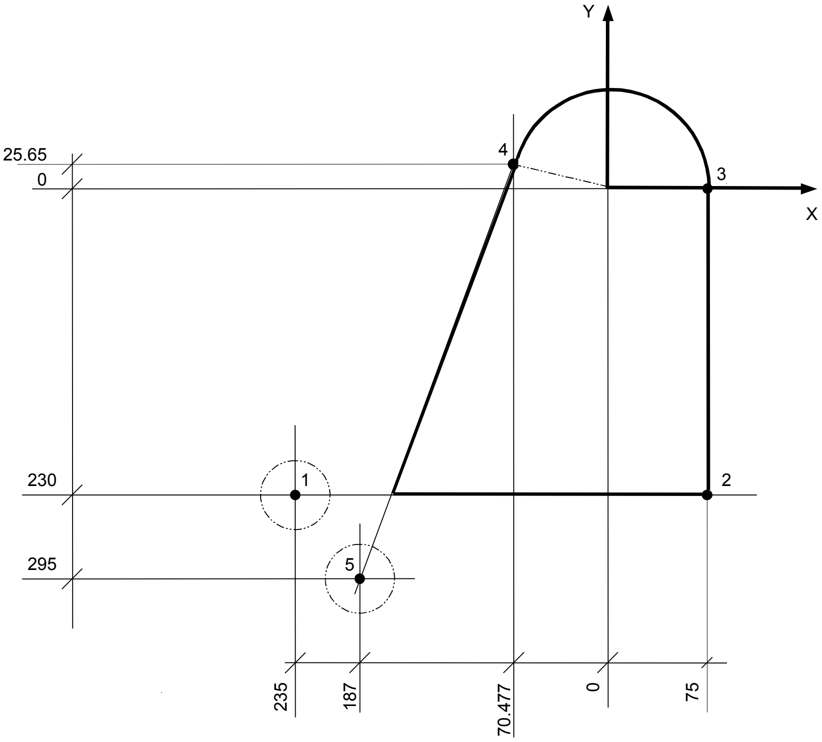
G27 and G28 differ only in the step with decelerated motion.
Positioning at the machining rate (G1, G2, G3) is available in continuous mode (G27, G28 and G29) whereas rapid positioning (G0) is always point to point, i.e. with deceleration down to null velocity and accurate positioning regardless of the system status.

With G27-G28 (continuous mode) the control explores and executes the profile as if it were a single block. For this reason, auxiliary functions M, S and T are not allowed within the profile executed in G27-G28.

Continuous mode can be temporarily closed by a G00 move that is still part of the profile. The allowed M, S and T functions may therefore be programmed in a block following G00.

NOTE:
The G code that has been configured in AMP (typically G27) is automatically selected at power-up or after a reset.

Example:
This is a contouring example in continuous and point-to-point mode.



Program 1 (continuous mode):

```
         (UGS,X,-400,100,Y,-400,100)
         N9 (DIS,"MILL DIA. 16")
         N10 T4.4 M6 S800
1        N11 G X-235 Y-230 M13
         N12 Z-10
2        N13 G27 G1 X75 F500          ;Continuous mode starts (G27)
3        N14 Y .
4        N15 G3 X-70.477 Y25.651 I J
5        N16 G1 X-187 Y-295
         N17 G Z5 M5                  ;Temporary shift to point to point mode
         N18 (DIS,"MILL DIA. 28")     ;for spindle stop, tool change and S functions
         N19 T5.5 M6 S1200
         N20 X.. Y.. M13
         N21 Z-..
         N22 G1 X.. Y..               ;Continuous mode restarts
```

IMPORTANT

If G29 were programmed in block N17, continuous mode would stop and subsequent moves in G1-G2-G3 would be performed in point-to-point mode.

Program 2 (point-to-point mode):

```
        (UGS,X,-400,100,Y,-400,100)
        N9 (DIS, "MILL DIA. 16")
        N10 T4.4 M6 S800
1       N11 G29 G X-235 Y-230 M13      ;Point-to-point operation starts
        N12 Z-10
2       N13 G1 X75 F500 M5             ;Spindle stop
3       N14 Y S1200 M13                ;Spindle CW with coolant
4       N15 G3 X-70.477 Y25.651 I J
        N16 DWT=2
5       N17 G1 G4 X-187 Y-295          ;Dwell at the end of the element
        N18 G Z5 M5
        N19 (DIS,"MILL DIA. 28")
        N20 T5.5 M6 S1200
        N21 G X.. Y.. M13
        N22 Z-..
        N23 G1 X.. Y..
```

IMPORTANT

By programming point-to-point with G29 in block N11, M and S functions have been included in the profile (blocks N13 and N14). The dwell at the end of the element (block N17), however, can also be programmed in continuous mode.

AUTOMATIC DECELERATION ON BEVELS IN G27 MODE

When G27 mode is active, the control automatically calculates the vector velocity on the bevels (i.e. between two subsequent moves) using a two-step algorithm.

During the first step the vector velocity is calculated with a formula based on profile variations. The variation of the profile is associated to the angle formed by two subsequent moves.

The control compares the actual angle with the MDA value; if the angle is greater, the vector velocity is put to zero as in G29 mode; otherwise, the control calculates for this bevel a velocity that is based on the angle, MDA and VEF values.

The second step of the algorithm, called "***look ahead"**,* is optional. It can be enabled or disabled according to the value of the DLA variable.

The "***look ahead***" step is an optimisation of the first step. In fact, in order to provide a correct stop at the profile end, the calculated vector velocity is re-processed taking into account the total distance to be covered in G27 mode and the acceleration configured for each axis.

IMPORTANT

The ***look ahead*** feature (G1 G27) does not handle feedrate override. In fact, at this stage a 100% feedrate is assumed. Higher feedrates may generate SERVO ERRORS.

DLA - Deceleration Look Ahead

The DLA code enables/disables look ahead calculation in G27 dynamic mode. The control reads the motion blocks that make up the profile and those that follow the block in execution in order to recalculate the exit feedrate for the various blocks. It also calculates the deceleration on the bevels according to the profile. If the profile includes sudden trajectory variations and there are not enough block lengths to ensure appropriate deceleration, it is critical for the system to anticipate these events so that velocities can be adjusted. The number of motion blocks the system can look ahead after the current block can be specified in the characterisation. It ranges from 2 to 64 blocks.

Syntax

DLA=*value*

---

where:

value can be: **0** disables look ahead
1 enables look ahead

NOTE:
If DLA=1 system block time increases because the control must execute a greater number of calculations for each instruction. This results in greater accuracy.

It is advisable to set DLA=0 when it is clear that the programmed feedrate and the total distance to be covered in continuous mode are such as to provide a good stop at the end of the profile.

With DLA=0 the control will consider only the deviations from the theoretical profile on bevels.

Characteristics:

The default value of this variable is configurable in AMP.

DYM - Dynamic Mode

The DYM defines the type of algorithm to use for calculating the velocity between one element and the next with G27 active.

Syntax

DYM=*value*

---

where:

value It is a numeric value which can be:

0 to use the standard 10 Series formula
1 to use the standard 8600 Series formula
2 to use the 1° alternative 10 Series formula
3 to use the 2° alternative 10 Series formula

The standard Series 10 algorithm is based on precise mathematical formulas which assume a linear response from the machine and that the dynamic parameters configured are always applicable under any condition.

The algorithm already present in the 8600 series uses approximate formulas, therefore applying greater restrictions on movement.

The alternative algorithms keep into consideration the dynamic components (acceleration) that the axes can bear when passing from one block to the other. In this way it recalculates the final speed at of the first block in order to pass to the next one without excessive stress on the machine.

It is recommended that you verify the behaviour of both algorithms on the machine and then decide which default algorithm best suits that particular machinery and that particular type of work.

Characteristics

The default value of this variable can be set in AMP.

MDA - Maximum Deceleration Angle

The MDA code defines the maximum angular axis departure in which G27 is active. The selected value (from 0 to 180 degrees) defines an angle that is the limit of G27 operation.

Syntax

MDA=*value*

---

where:

value is a numeric value with the following characteristics:

- angle between 0° and 180° if DYM = 0
- number between 0 and 2 if DYM = 1

In both cases, it represents the maximum deviation between two consecutive elements beyond which, in G27, the stop is forced on the final point.

If DYM = 1, the value is to be calculated as:

sine of the maximum angle for deviations ≤ 90°

1 + [sine (angle - 90°)] for deviations > 90° and ≤ 180°.

- Not significative if DYM = 2 or DYM=3

Characteristics:

In order to alter the default value (MDA = 90 degrees) you may assign MDA in the configuration or enter it through a specific data entry or a part program block.

IMPORTANT

The system forces the axis to decelerate to zero velocity when the direction is greater than the angle defined by the MDA value. The system calculates a deceleration ramp for the programmed axis if the direction is less than or equal to the angle defined by the MDA value.

Since the system calculates deceleration on bevels from the actual angle and the MDA and VEF values, it is possible to alter velocity reduction by changing the MDA value. Small values of MDA generate dramatic deceleration on bevels.

The system RESET restores the configured MDA value.

Examples:
DYM=0
MDA=90°
MDA=180°

DYM=1
MDA=1
MDA=2

VEF - Velocity Factor

The VEF code defines a velocity determining factor on bevels in the G27 mode. The velocity calculated from the MDA value can be increased or decreased by changing the VEF value. Small VEF values dramatically reduce velocity on bevels.

Syntax

VEF=*value*

---

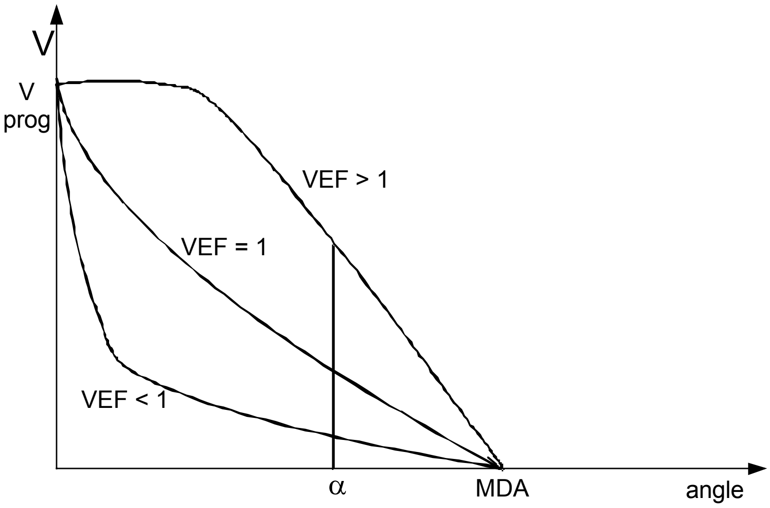
where:

value	is a number with the following characteristics:		
	• number from 0.1 to 8	if DYM = 0	The default value is 0.8.
	• number from 0 to 99999	if DYM = 1	The default value is 0.8.
	• number from -1 to 99999	if DYM = 2	The default value is 0.8
	• number from 0 to 99999	if DYM = 3	The default value is 0.8

Characteristics:

The characteristics of the velocity calculation vary according to the value of the DYM variable.

DYM = 0
The following diagram shows different decelerations calculated by the system by varying the VEF value and keeping the MDA value constant.



where:

V	is the velocity on the bevel
α	is the angle between two subsequent movements
Vprog	is the programmed feedrate

DYM = 1

The VEF code defines the maximum form error admissable on the bevel. If the value is 0, at the end of each block the system deccelerates the axes to zero.

DYM = 2

The VEF value defines the maximum speed "step" for the axis in the passage from one block to the next: for example if VEF=0.8 the axis will have a speed "step" of 1+0.8 of the acceleration of set working acceleration; if VEF=-0,3 the axis will have a speed "step" of +1-0.3 or +0,7.
The system will calculate the speed on the edges according to all the axes that are part of the movement; each axis will have a different speed and the system will choose the minimum among these.

DYM = 3
VEF is a value that defines the time, in ms, it takes to reach the speed to be applied to the corner as a function of the accelerations configured on the various axes. The formula to be applied is:

Vcorner = Acceleration * VEF

Hence, having determined Vcorner, the VEF to be programmed will be:

$$\text{VEF} = \frac{\text{Vcorner}}{\text{Acceleration}}$$

The system will calculate the speed on the edges according to all the axes that are part of the movement; each axis will have a different speed and the system will choose the minimum among these.

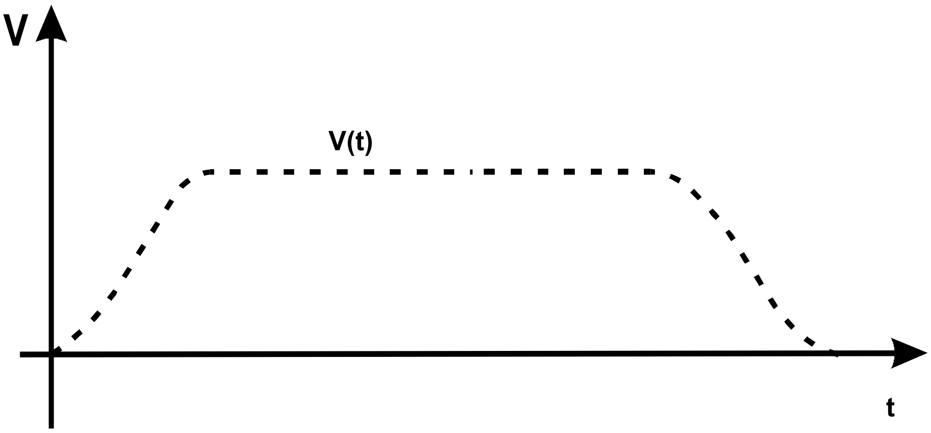
IMPORTANT A system RESET restores the VEF value configured in AMP

Jerk Limitation

The speed diagrams shown in the previous sections show the continuity of the speed function **V(t)**, while the acceleration function **a(t)** has a step pattern. Depending on the characteristics of the machine and the type of machining process, this may cause defects in the finish of the part.

This problem may be solved using an acceleration function **a(t)** with a continuous pattern.

The purpose of the "Jerk Limitation" function is to limit variations in acceleration, so as to control its maximum value, resulting in smoother movement and, consequently, a better surface finish.



MOV - Enable Jerk Limitation

The MOV code is used to define some characteristic of the movements management.

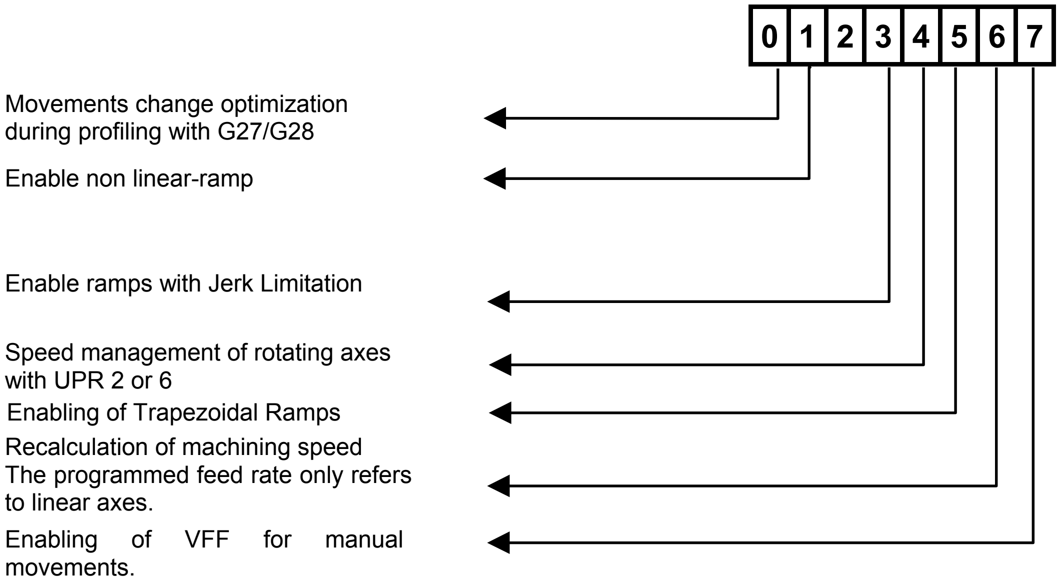
Syntax:

MOV = *value*

---

where:

value movements behaviour to be enabled.
The value to be specified is obtained from the sum of the decimal weights corresponding to each of the features desired.


0
1
2
3
4
5
6
7
Movements change optimization during profiling with G27/G28
Enable non linear-ramp
Enable ramps with Jerk Limitation
Speed management of rotating axes with UPR 2 or 6
Enabling of Trapezoidal Ramps
Recalculation of machining speed The programmed feed rate only refers to linear axes.
Enabling of VFF for manual movements.

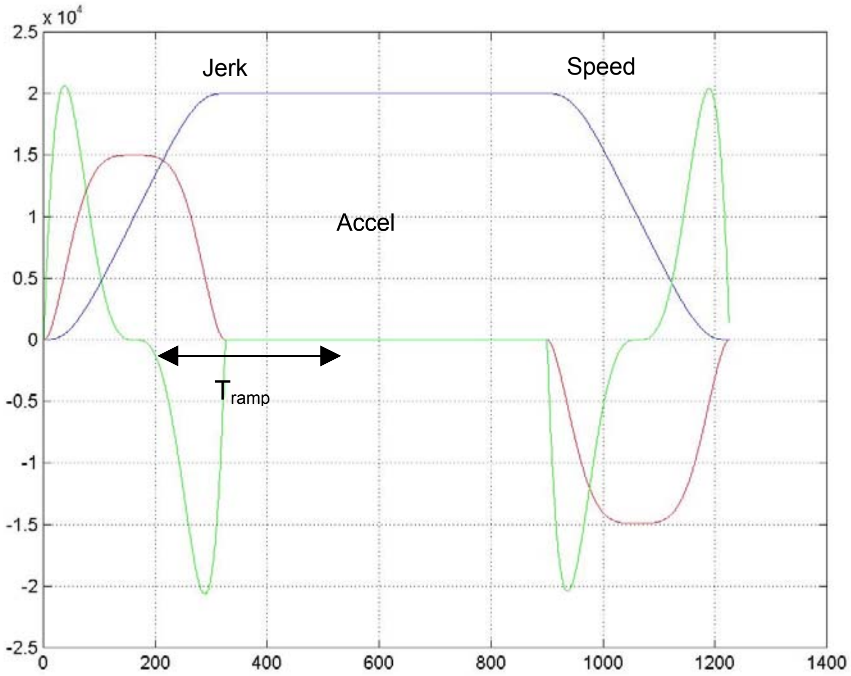
The default value of this variable is 0. The MOV value can be configured in AMP. The RESET restores the default value.

Example:

if you want to use Jerk Limitation with non-linear ramps and at the same time have VFF enabled for normal movements, the MOV variable must be set to the value 128+8=136

Meaning of bits 1, 3 and 5:

Bit 1 (value 2): Enables the non linear ramps (**S ramps**) where the execution mode is characterised by the value of the JRK variable. The system modulates continuously the acceleration value between 0 and a maximum whose value is given by A/JRK, so that, with JRK=1 the nominal acceleration value is reached, with JRK=2 half the nominal acceleration value is reached, with JRK=0.5 twice the nominal acceleration value is reached. The ensuing jerk values on the movement will NOT be controlled and will range from 0 to an undetermined value.



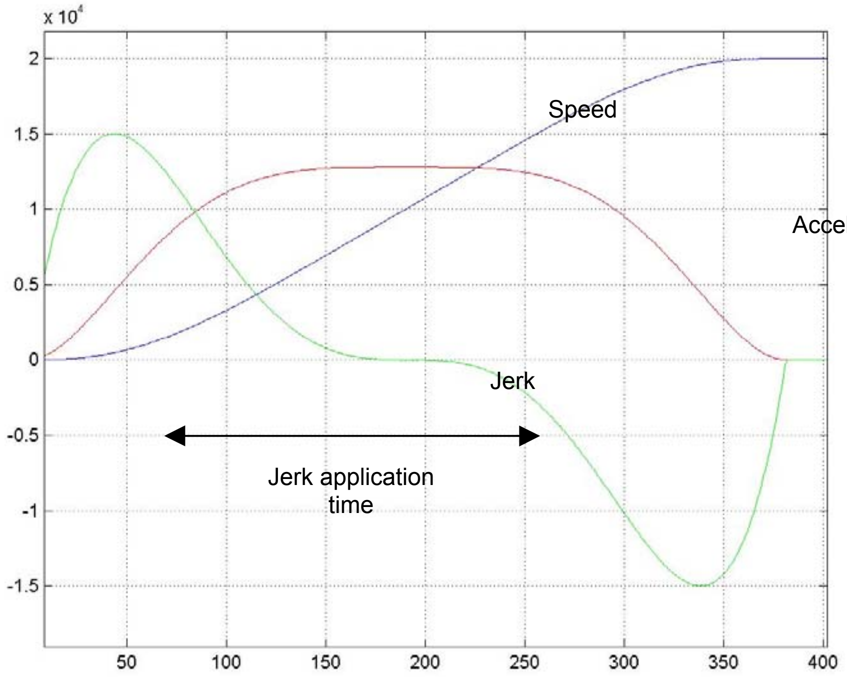
In the example above, a movement at a speed of 20000 mm/min with an acceleration of 1500 mm/sec^2 has been programmed. The maximum acceleration value is reached only at a time halfway between the acceleration stage or the deceleration stage. For JRK=1 the acceleration value reached corresponds to the value requested.

The execution time of the acceleration stage corresponds to the time it would have taken using a linear acceleration ramp, multiplied by the value of JRK and constant 1.485.

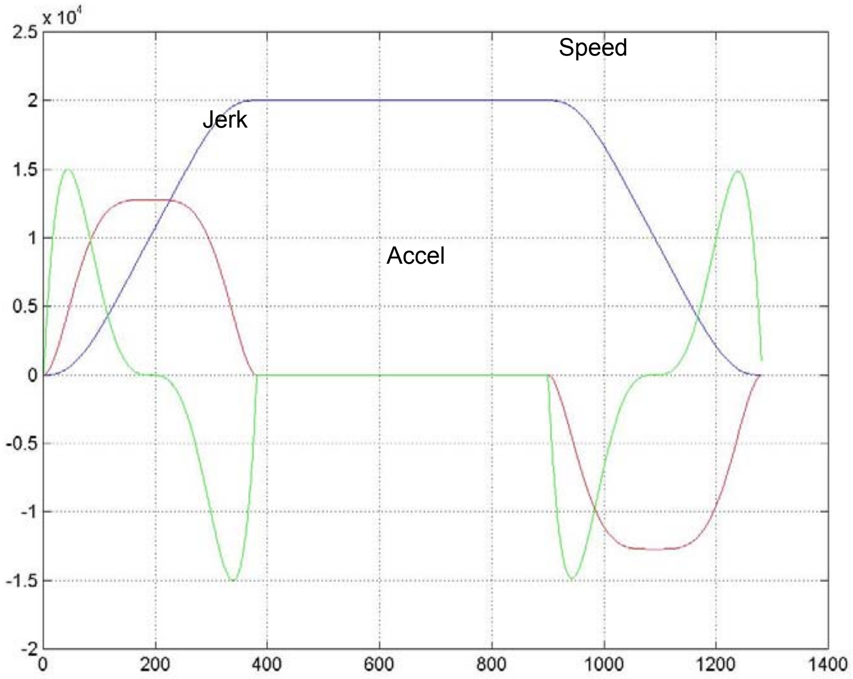
$T_{ramp} = T_{linear\ ramp}$ * JRK * 1.485.

Ramp time is also affected by the “minimum ramp time” parameter configured in AMP for each axis. If the time calculated for the ramp does not meet the minimum ramp time limit, the ramp will be recalculated to comply with this requirement.

N.B. The minimum ramp time is time it takes to reach the maximum acceleration value (Jerk application time) so that, as can be seen from the chart below, the total ramp time corresponds to twice the minimum time.



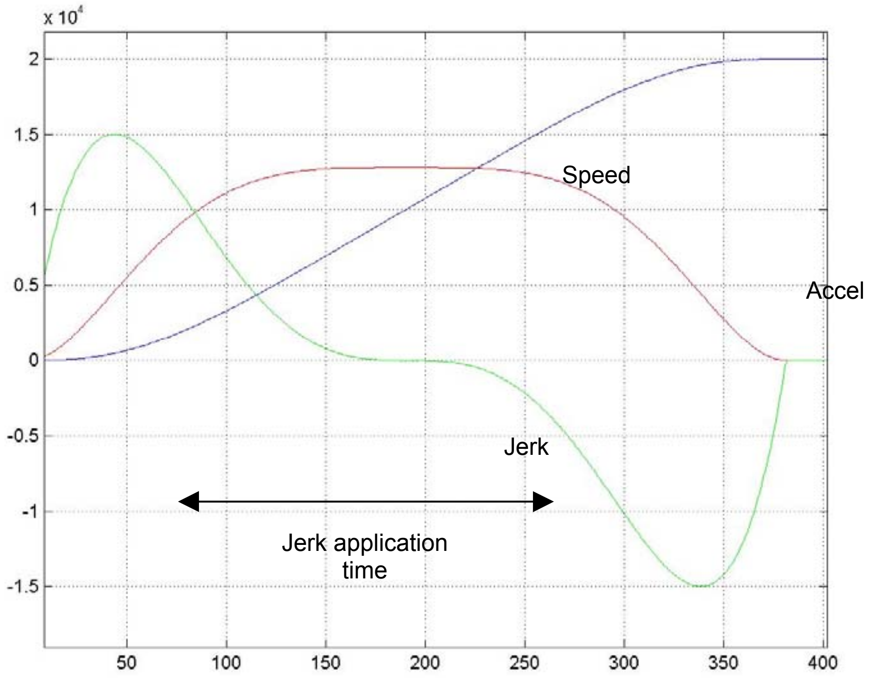
Bit 3 (value 8): Enables the non linear ramps (**ramps with jerk limitation**). Acceleration ramps are calculated on the basis of the jerk parameters configured in AMP. The jerk value used within a movement is calculated by the system so as to ensure that the movement complies with the jerk characteristics specified for each axis. The jerk value configured for each axis must be construed as the maximum value that can be reached during the acceleration stage. The system modulates the jerk value continuously between 0 and its maximum value, and it will modulate in the same manner the acceleration value, which will not necessarily reach the maximum value configured.



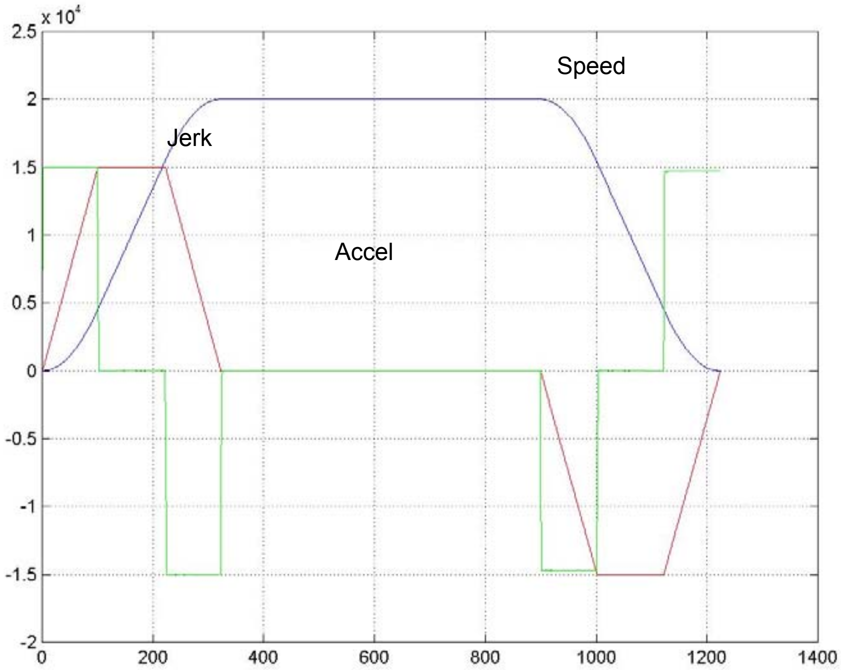
In the example above, a movement at a speed of 20000 mm/min with an acceleration of 1500 mm/sec^2 and with jerk of 15000 mm/sec^3 has been programmed. The maximum acceleration value is reached only at a time halfway between the acceleration stage or the deceleration stage and it is below the maximum value that is requested. Only with a higher jerk it will be possible to reach the maximum acceleration.

Ramp time is affected by the jerk parameter and by the “minimum ramp time” parameter configured in AMP for each axis. If the time calculated for the ramp does not meet the minimum ramp time limit, the ramp will be recalculated to comply with this requirement.

N.B. The minimum ramp time is time it takes to reach the maximum acceleration value (Jerk application time) so that, as can be seen from the chart below, the total ramp time corresponds to twice the minimum time.


x 10⁴
2
1.5
1
0.5
0
-0.5
-1
-1.5
50
100
150
200
250
300
350
400
Speed
Accel
Jerk
Jerk application time

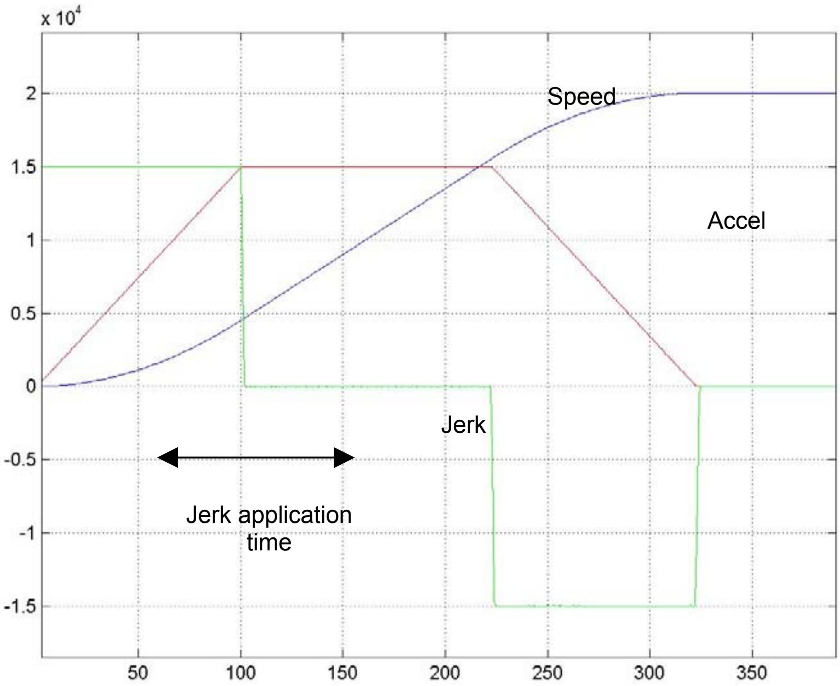
Bit 5 (value 32) : Enables the non linear ramps (trapezoidal ramps). Acceleration ramps are calculated on the basis of the jerk parameters configured in AMP. The jerk value used within a movement is calculated by the system so as to ensure that the movement complies with the jerk characteristics specified for each axis. The jerk value configured for each axis must be construed as the value that can be used during the acceleration stage. The system keeps the jerk value continuously during the acceleration stage.



In the example above, a movement at a speed of 20000 mm/min with an acceleration of 1500 mm/sec^2 and with jerk of 15000 mm/sec^3 has been programmed. The maximum acceleration value is reached and maintained for a certain time in the central part of the acceleration or deceleration stage; it may or may not reach the maximum value requested as a function of speed leap to be made.

Ramp time is affected by the jerk parameter and also by the “minimum ramp time” parameter configured in AMP for each axis. If the time calculated for the ramp does not meet the minimum ramp time limit, the ramp will be recalculated to comply with this requirement.

N.B. The minimum ramp time is time it takes to reach the maximum acceleration value (Jerk application time).



Characteristics:

Non linear ramps work both on point to point movements (G29) and continuous movements (G27 and G28). The activation or deactivation of non linear ramps in a continuous movement is not perceived, it is only perceived in point by point movement mode. Similarly, it is not permitted to change from G27 to G28 (and vice versa) if non linear ramps are active; in such circumstances it is necessary to close the continuous movement (G29) and open another according to the new modality.

It is recommended to use value 8 for machines with high dynamic levels (high jerks), while value 32 should be used in machines where dynamic levels have to be reduced (low jerks). Configured jerk being the same, value 32 performs better (in terms of execution times) than value 8, but engenders more “abrupt” movements.

Meaning of bit 0:

Bit 0 (value 1): Movement change optimization during profiling with G27 and G28. This option must be used particularly for curves defined “by point”. It allows a better management of the profile curve variation (especially if defined with very short movements) avoiding axes movements out of the profile. This option is activated by default in the case of non linear ramps.

Meaning of bit 4:

Bit 4 (value 16): enables the type of rotary axis speed management mode to be adopted with UPR 2 or 6 active; if this bit = 0, standard management is maintained; if this bit = 1, the rotary axes will use different speed diagrams.

In the former case, the positioning of the rotary axes conforms to the transformation requested by the UPR, but the particular computation algorithms employed may cause appreciable showdowns around singular points.

In the latter case, slowdowns are eliminated, but the positioning of rotating axes may fail to conform to the transformation requested by the UPR: in such cases, the axes will lag behind. Since this occurs around singular points (mostly with vertical tool axis), the rotary axis positioning delay has virtually no effects on processing results.

Meaning of bit 6:

Bit 6 (value 64): defines that the feed rate programmed in a movement block refers to linear axes. The programming of a rotary axis, in addition to the linear axes, entails an automatic recalculation of the speed so that it remains the same along the linear axes. The recalculation of the speed is only applied if both the linear axes and the rotary axes are present in a movement. It is not applied in the case of circular movements (G2/G3) in which a rotary axis is part of the interpolation plane.

IMPORTANT

This feature **must NOT** be activated if speed programming is executed in G93 or with t. In this case, the resulting machining time would not be correct.

Meaning of bit 7:

Bit 7 (value 128): enables the VFF algorithm also for manual movements that are normally executed using the "tracking error" algorithm only.

JRK - Jerk Time Constant

The JRK code defines the acceleration management mode during the execution of movements with S ramps (MOV=2).

Syntax:

JRK = *value*

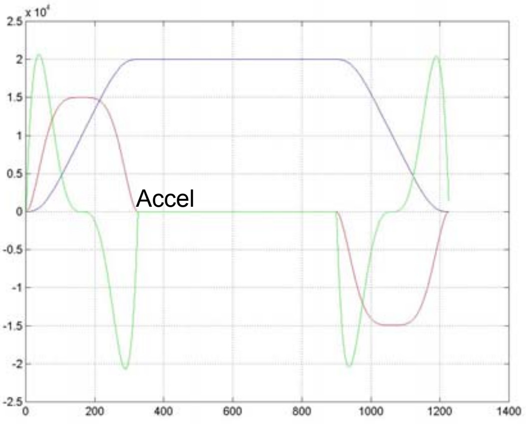
---

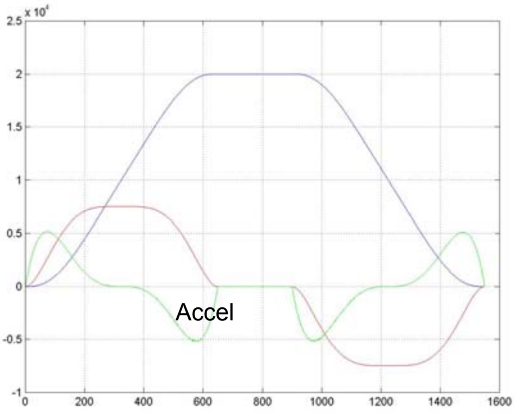
where:

value	a numeric value greater than 0.5, which is used to define the acceleration management mode in a non-linear ramp

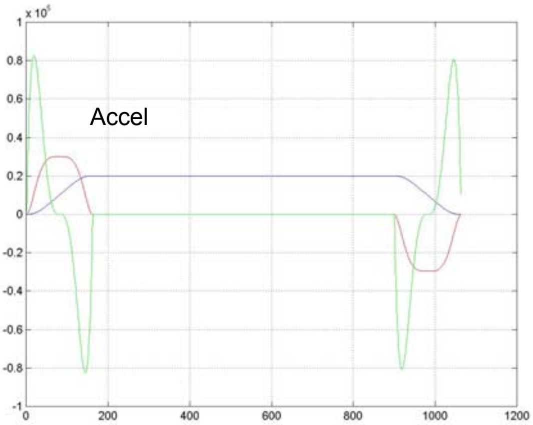
Characteristics:

The default value is 1. The JRK value can be configured in AMP. The RESET command restores the default value. By setting JRK = 1, the acceleration ramp retains the values configured for axis accelerations. Acceleration increases with a value lower than 1, and decreases with a value higher than 1.





It is necessary an acceleration of 1500 mm/sec² in both movements. In the first movement has been used JRK=1 so that the nominal acceleration value is reached, in the second case has been used JRK=2 so that half the nominal acceleration value is reached. In the next example has been used JRK=0.5 so that twice the nominal acceleration value is reached.



JRS - Jerk Smooth Constant

The JRS code defines the threshold value for limiting the speed with Jerk Limitation activated.

Sintyax:

JRS = *value*

---

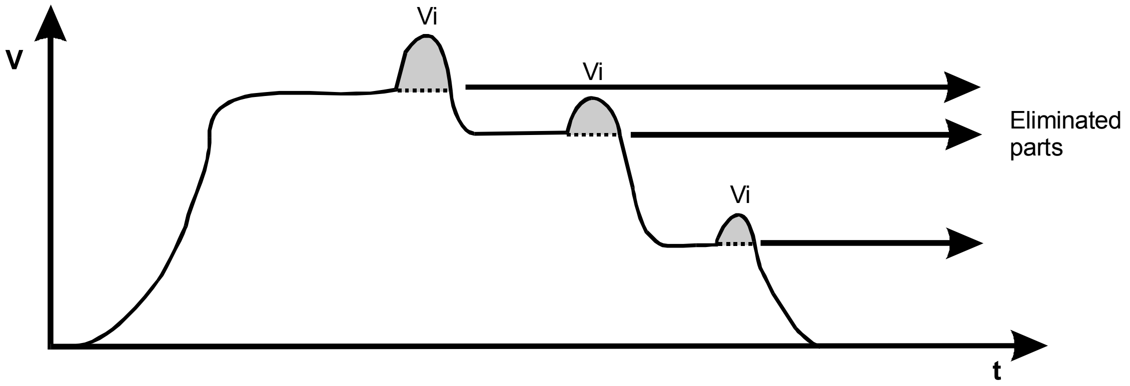
where:

value	is a a numeric value greater than 0, which used for defining a speed threshold below which the speed is to be limited if the programmed speed cannot be reached.

Characteristics:

A characteristic of the lookhead algorithm for Jerk Limitation is that it avoids continuous acceleration and deceleration that would cause oscillation in the movement of the axes. This could happen if the if the programmed axes do not allow the speed set **(Vi)** to be reached.

To this aim, the speed diagram is "cut" as illustrated in the figure below:



To prevent the "cut" part from being too large and considerably increasing the machining times, a threshold below which the speed is to be limited is defined.
This threshold is calculated as a function of the JRS parameter on the basis of the following rule:

Vmax	The maximum speed value calculated by the algorithm on the movements taken into consideration
Vmin	The minimum speed value calculated on the movements taken into consideration.

The system checks whether

$$| Vmax - Vmin | < Vmin \bullet JRS$$

If it is, the Vmax is set equal to Vmin, otherwise it is recalculated so that there is a section with a constant speed in the upper part of the ramp.

IMPORTANT

The default value is 1. The RESET command restores the default value.

ODH - Online Debug Help

This system variable allows to check whether the part program includes continuous moves (G28) in which the velocities programmed on the profile are too high and may generate path errors.

Syntax

ODH=*value*

Characteristics:

To obtain reliable results from the use of ODH, the variable itself must be reset before it is used in the part program: that is, ODH must be forced to zero before the blocks to be tested. This variable is normally used when the Part Program is debugged: the system modifies the ODH variable when it detects critical conditions.

This information is coded in bits with the following meanings:

Bit 0: Indicates that the queue of the elements processed continuously is too short. It occurs when, in G28 (and in some cases in G27), the machining speed is too high with respect to the value set in AMP as the number of blocks to be processed during continuous motion. The possible solutions are to decrease the speed or increase the size of the queue of precalculation elements.

Bit 1: Indicates an excessively high speed and occurs when there are too short elements in the profile, which are skipped due to the excessively high speed. In this case, the machined profile is deformed. The possible solutions are to reduce the speed, or to recalculate the points in the program. To avoid profile deformation, we recommend that bit 0 (MOV = 1) be activated in the MOV variable.

Bit 2: Occurs when an excessively high acceleration is requested in G26 with respect to the one set, in moving on from one movement element to the next

Bit 3: Occurs when the system has forced a brake to avoid deforming the profile when it cannot completely calculate the acceleration/deceleration ramps for the movement elements analysed, during machining with non-linear ramps enabled.

Example:

```
. . . . . . . .
. . . . . . . .
ODH=0                              ;the variable is taken to zero for velocity verification
G1G28F1000                         ;continuous mode starts
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .                    ;continuous mode blocks
. . . . . . . .
. . . . . . . .
. . . . . . . .
G29M5                              ;continuous mode end
(GTO,KO,ODH=1)                     ;branches if ODH=1
(DIS;"VEL.WITHIN SYSTEM LIMITS")
(GTO,CONTINUES)
. . . . . . . .
"KO"
(DIS,"VELOCITY TOO HIGH")
. . . . . . . .
. . . . . . . .
"CONTINUES"
```

MBA – MultiBlock retrace Auxiliary functions

The MBA (Boolean) variable makes it possible to enable/disable the auxiliary functions (M functions) during Retrace operations:

Syntax:

MBA = *value*

Characteristics:

The emission of auxiliary functions is performed only during forward retrace

(see user manual)

value = 0 Emission disabled

value = 1 Emission enabled

IMPORTANT

After a reset, the variable is reset with the value configured in AMP.

REM – Automatic return to profile at end of move

The REM (Boolean) variable determines the Jog return mode.
If the variable is set on 0 the return point is the point at which the hold stage begins (standard execution).
If the variable is set on 1 the return point is the final point of the movement in which the hold stage begins. The return movement is performed automatically, without manual movements (JOG).

Syntax:

REM = *value*

---

value = 0 Return to input point in hold

value = 1 Return to input point at end of movement

IMPORTANT Reset does not modify the value of the variable.

IPB (DTL) - In Position Band

The IPB command lets you define the axes positioning tolerance.

Syntax

(IPB, *axis1*,[*axis2*,**. . .**,*axis9*]**)**

---

where:

axis1 . . axis9	Is an axis address and a numerical value. In the IPB command you can program up to nine axes, each of which must be configured in the system. You should program tolerance dimensions in the measuring unit (G70/G71) that is active when the IPB command is executed.

Characteristics:

This value specifies the tolerance between the theoretical and the actual axis positions at motion end. In G00 positioning mode, the axis must be within this tolerance before the next block is executed.

If you program 0 in the IPB command, the control uses the default positioning tolerance specified in the characterisation.

If no axes are specified, the control uses the positioning tolerance that is currently active for that axis.

You cannot specify the same axis twice in one IPB command.

The IPB command causes an error when the following conditions/modes are active:

- Cutter diameter compensation (G41-G42)
- Continuous mode (G27-G28)
- In internal computations the value entered is rounded out to < 1 digit.
  The error generated is NC 133.

Before programming an IPB command you must disable these codes from program.

IMPORTANT

The programmed value overrides the one configured in AMP. A RESET does not restore the configured value.

Example:
(IPB, X 0.1, Y 0.05)
This block specifies a positioning tolerance of 0.1 for X and 0.05 for Y.

G70 G71 - Measuring Units

The G70 and G71 codes define the measuring unit used by the control.

G70	specifies inch programming
G71	specifies millimetre programming

Syntax

G70 [*G-codes*] [*operands*]

G71 [*G-codes*] [*operands*]

---

where:

G-codes	Other G codes that are compatible with G70 and G71 (See "Compatible G codes" table in Chapter 1).
operands	Any operand or code allowed in a G function block.

Characteristics:

If neither G70 nor G71 is programmed, the system will assume the default measuring unit stored in the system configuration (typically G71).

When the system switches from G71 to G70 or from G70 to G71 it also converts all position and feedrate information into the relevant unit.

IMPORTANT

However, offset and origin tables are not automatically converted into the alternative unit when the system switches between G71 and G70.

Shifts between G70 and G71 do not affect the values read from probing cycles G72, G73 and G74.

G90 G91 G79 - Absolute, Incremental and Zero Programming

These G codes define whether programming dimensions are absolute, incremental or referred to the machine zero.

G90	Sets absolute programming, i.e. moves referred to the current origin (position to move).
G91	Sets programming in the incremental system, i.e. moves referred to the position reached with the previous move.
G79	Sets programming in absolute reference to home position. It is valid only in the block in which it is programmed (distance to move).

Syntax

G90 [*G-codes*] [*operands*]
G91 [*G-codes*] [*operands*]
G79 [*G-codes*] [*operands*]

where:

G-codes	Other G codes that are compatible with G90, G91 and G79 (See "Compatible G codes" table in Chapter 1).
operands	Any operand or code that can be used in G function blocks.

Characteristics:

If none of these codes is programmed, the default programming mode is absolute or G90, i.e. coordinates referred to the programmed origin.

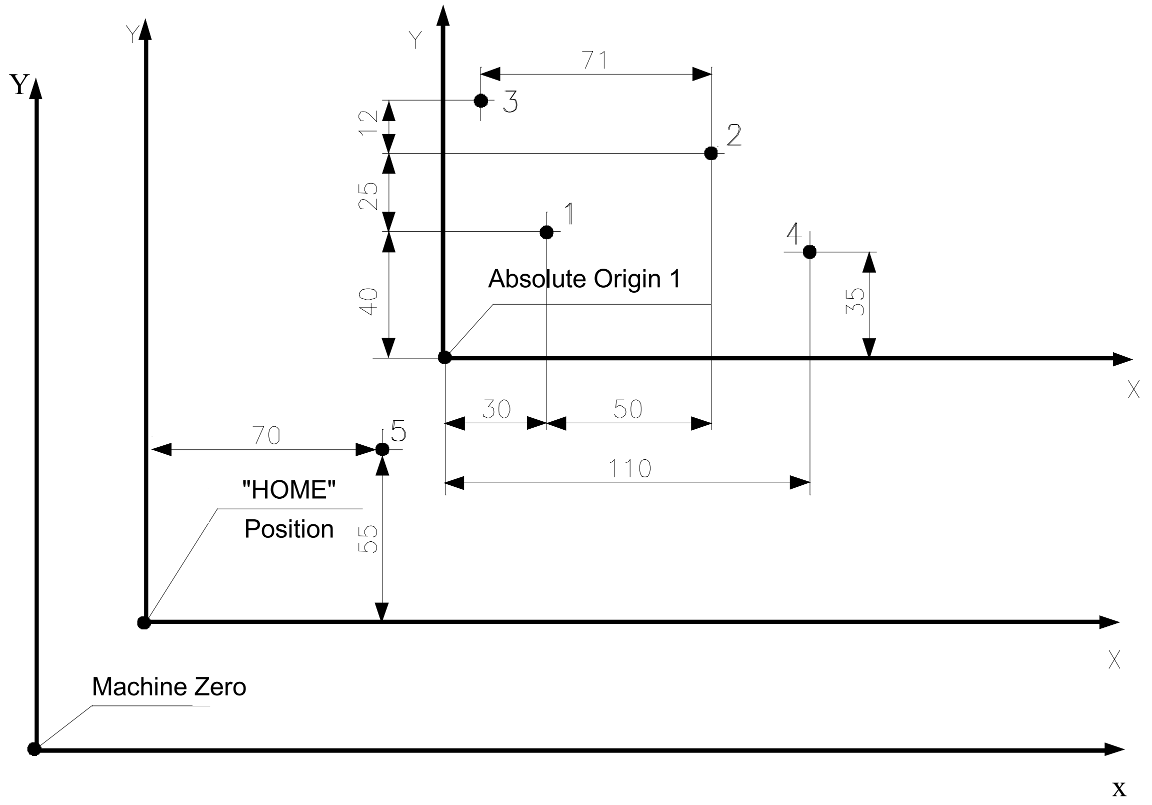
G90 and G91 are modal whereas G79 is not. After programming a block with G79, the control restores the programming mode of the previous block.

Using characters >> a mixed incremental/absolute programming in the same block is also possible.

Characters >> positioned before the numeric value of an operand, indicate that it must be considered as an incremental value and that it is valid for that operand only. Characters >> have a meaning only if G90 absolute programming is active. They may be used for all operands on which it is possible to use G91 function.

Example:
G90 G1 X + 80 Y >> + 35 Z-70
The value associated to Y must be considered as incremental.

Example:
This example shows how to use the different reference systems: absolute, incremental and HOME position.



Program:

```
(UGS,1,X,-50,100,Y,50,100)
(UAO, 1)                  ;Enables absolute origin 1
N1 G X Y                  ;X and Y positioned to absolute origin 1 (assuming default
                          mode G90)
N2 X30 Y40                ;X and Y positioned to point 1
N3 G91 X50 Y25            ;Incremental mode positioning to point 2 ( X+50, Y+25 from
                          point 1)
N4 X-71 Y12               ;Incremental mode positioning to point 3 ( X-71, Y+12 from
                          point 2)
N5 G90 X110 Y35           ;Absolute mode positioning to point 4 ( X+110, Y+35 from
                          the origin)
N6 G79 X70 Y55            ;Positioning referred to home position on point 5 (X+70,
                          Y+55 from home position)
```

G92 G98 G99 - Axis Presetting

G92 and G98 represent an alternative method for introducing axis offsets. Since G92/G98 define the reference position, they are used in a part program block by themselves. The difference between the actual and the new position is stored in a separate G92/G98 offset register. In this way, other active offsets such as tool offset and origins, will not be destroyed when a G92/G98 offset is introduced. The G99 code resets the G92/G98 code.

Syntax

G92 *axes*
G98 *axes*

where:

axes	Is the number of axes (6 axes max.)

Characteristics:

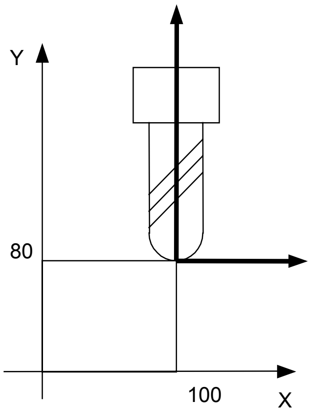
G98 works the same way as G92, save that G92 does not take into account the MIRROR applied to the programmed axes and G98 does.

Codes G92 and G98 are cancelled by the following functions:

- G99
- M2
- M30
- System reset
- PLUS

Active reset has no influence on a offset programmed with G92 or G98. The G92 or G98 offset shifts the origin for a part program but does not cause any axis motion. When the axis value is entered in a G92 or G98 block it becomes the current axis position.

Example:



```
. . . . . . . .
. . . . . . . .
G0 X100 Y80
G92 X0 Y0
. . . . . . . .
. . . . . . . .
```

G04 G09 - Dynamic Mode Attributes

Two G codes belong to this class:

G04	Dwell at end of block
G09	Deceleration at end of block

Syntax

G04 [*G-codes*] [*operands*]
G09 [*G-codes*] [*operands*]

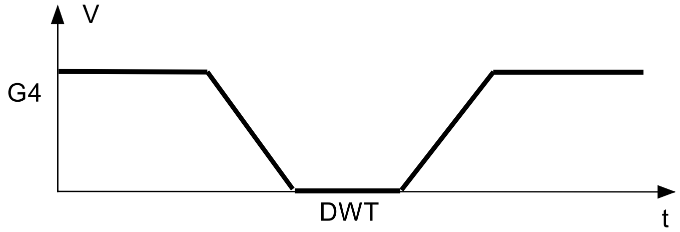
where:

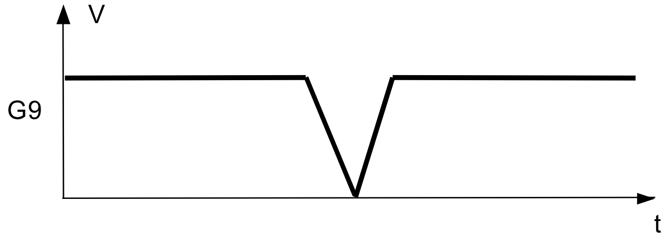
G-codes	Other G codes that are compatible with G4, G9 (See "Compatible G codes" table in Chapter 1).
operands	Any operand or codes allowed in a G function block.

Characteristics:

G04 causes a dwell at motion end in a block. The DWT command must be programmed in a block that precedes G04. If no DWT is be programmed, the system assumes the characterised dwell.
G04 is allowed only when the control is in point-to-point (G29) and is valid only in the block in which it is programmed. The value set in DWT may be expressed in seconds (G94 or G95 with G0 active) or revolutions (G95).

G09 forces a feedrate equal to zero at the end of the block in which it is programmed, but does not vary the machining status of the profile in progress. G09 does not cause any change in the control status and is valid only in the block in which it is programmed.





t - Block Execution Time

Both in G93 and G94 it is possible to establish the block execution time by programming the **t** function at the end of the block.

Example:
G1 X10 Y1 t6

Characteristics:

The **t** function is valid only for the block in which it is programmed. The time is expressed in seconds and the control automatically calculates the feedrate at which the moves of the axes programmed in the block will be executed.

DWT (TMR) - Dwell Time

The DWT command lets you assign a dwell time at block end. This dwell time is used by G04 and canned cycle blocks.

Although the DWT command can be programmed anywhere in the part program, but must come before any G04 or fixed cycle linked to it.

Syntax

DWT = *value*

---

where:

value	Is a time in seconds or revolutions. It can be programmed directly with a decimal number or indirectly with E parameters.

Example:

DWT	= 12.5	assigns a dwell time of 12.5 seconds
E32	= 13.4	assigns the value 13.4 to the E32 variable
DWT	= E32	assigns a dwell time of 13.4 seconds

G93 - V/D Feedrate

G93 defines the axes feedrate (F) as the reciprocal of the time in minutes required to execute the entity:

$$F = \frac{1}{T}$$

Linear interpolation:

$$F = \frac{Feedrate}{Distance}$$

Circular interpolation:

$$F = \frac{Feedrate}{Arc}$$

where:

Feedrate	linear or circular speed expressed in mmpm (G71) or ipm (G70)
Distance	vectorial distance of linear motion expressed in inches or mm
Arc	arc length expressed in inches or mm.

IMPORTANT The F word is mandatory in blocks where G93 is active.

Example:

```
G93 G1   X. . Y. . F. .
         X. . Y. . F. .
```

VFF - Velocity Feed Forward

This command enables/disables VFF. Il VFF has an effect on the interlocking of the axes allowing the speed to be controlled as well as the position.

Syntax

VFF=*value*

---

where:

value	Can be: **0** disables VFF: (only the position of the axes is controlled, that is taking into consideration the error margin between the tracking and its theoretical position). **1** enables VFF: (the speed of the axes is also controlled).

NOTE:
The VFF default value, 1, is configured in AMP.

CODES THAT MODIFY THE AXES REFERENCE SYSTEM

The commands in this class let you change the cartesian reference system in which the profile is programmed.

COMMAND	FUNCTION
SCF	Scale factor
MIR	Mirror machining
ROT	Active plane rotation
UAO	Absolute origin
UTO	Temporary origin
UIO	Incremental origin
RQO	Origin requalification

When active, the following functions must be programmed in this sequence:

SCF - G70/G71 - MIR - ROT - ORIGINS.

SCF - Scale Factors

The SCF command assigns a scale factor to programmed axis dimensions. The control applies the scale factors to the axes specified in the SCF command.

Syntax

(SCF [*axis1*, . . . , *axis9*]**)**
or
(SCF [,*value*]**)**

where:

value	Programs the scale factor. You can program the scale factor directly with a decimal number or indirectly with an E parameter.
axis1 . . . *axis9*	Addresses of axes configured in the system and scale factor.

Characteristics:

You can specify up to nine axes in the SCF command. The control cancels scaling for axes that are not specified in the command. A SCF command programmed without a scale factor or axes cancels scaling for all axes.

Example:

.
.
.

(SCF, 3)	Applies a scale factor of 3 to the programmed dimensions for all configured axes.

.
.

(SCF, X2.5, Y3)	Applies a scale factor of 2.5 to the programmed dimensions of the X axis and 3 to the Y axis and deactivates scaling for all other axes.
(SCF)	Deactivates scaling for all axes.

IMPORTANT The system RESET disables the scale factor for all axes

MIR - Using Mirror Machining

The MIR command reverses (mirrors) the programmed direction of motion for the axes you specify in the command referred to the current origin.

Syntax

(MIR ,[*axis1*, **. . . ,** *axis*9])

---

where:

axis1. . .*axis9* Is the name of the axis to be moved in mirror.

Characteristics:

You can program up to nine axes in the MIR command. Program an axis only once per each MIR command. If a given axis is not programmed in MIR, any mirror function for that axis will be turned off.

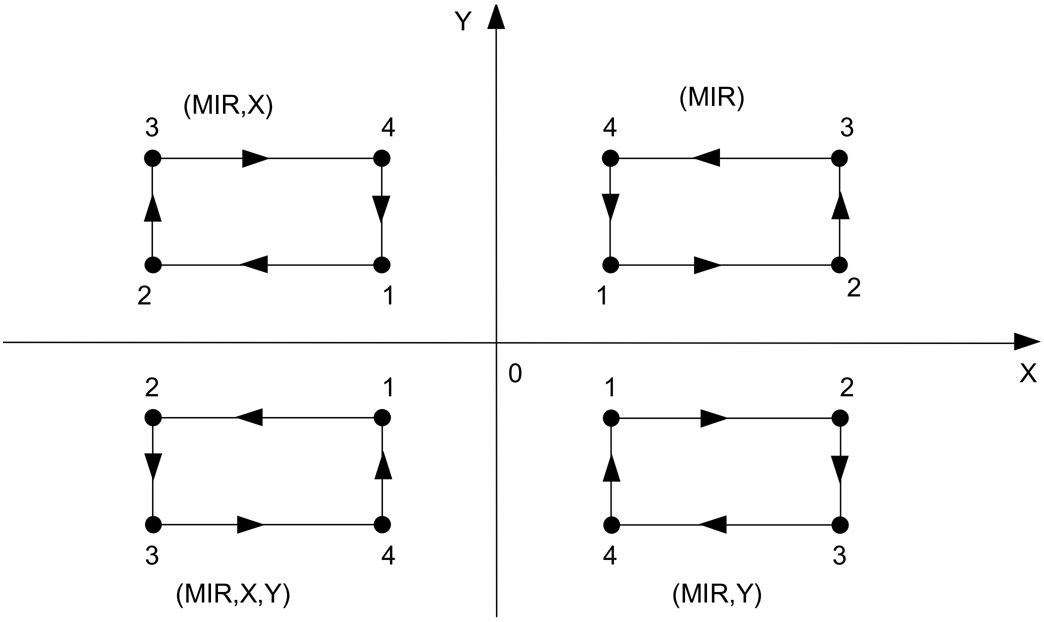
If no axis is programmed in the MIR command, the mirror function will be reset for all configured axes.

The control mirrors the programmed axis move with respect to the current origin, applying the mirror function from the first motion block including that axis after the MIR command.

If you use plane rotation (ROT) in conjunction with the mirror (MIR) command, the control processes them in the following order: MIR then ROT.

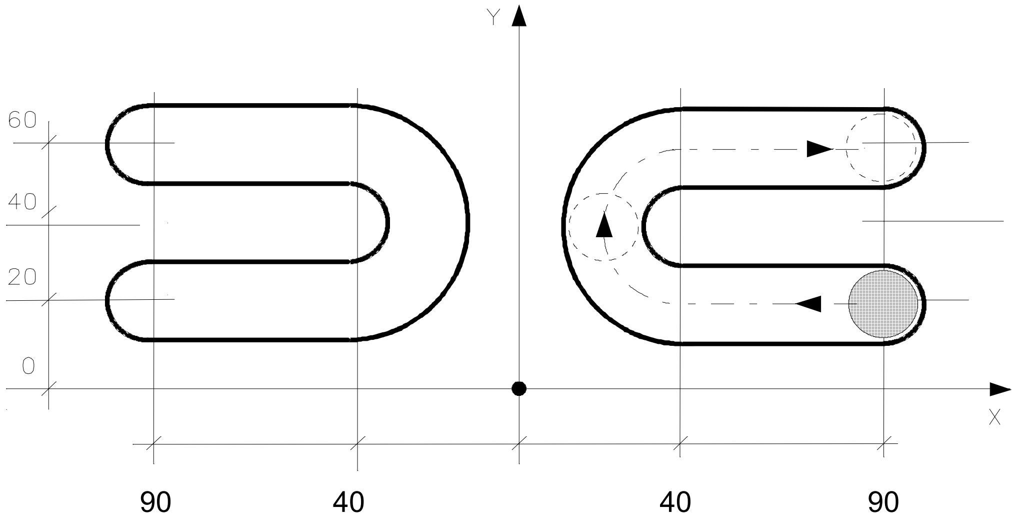
IMPORTANT The system RESET disables the MIR command for all axes. It is equivalent to programming (MIR) without parameters.

Example 1:
The following example shows how mirror machining works.



.	Mirroring not active. Moves occurring the first quadrant. Moves referred to the current origin.
.	
N24 (MIR,X)	Mirroring active for the X axis only. Programmed +X moves generate a move shown in the 2nd quadrant.
.	
.	
.	
N42 (MIR, X, Y)	Mirroring active for the X and Y axes. Result of programmed moves shown in the 3rd quadrant.
.	
.	
.	
N84 (MIR, Y)	Mirroring active for the Y axis and inactive for the X axis. Moves in the 4th quadrant.
.	
.	
.	
N99 (MIR)	Mirroring inactive for all axes. Moves in the 1st quadrant.

Example 2:
This example shows how to use the MIR command. Also note the use of RPT and ERP.



Program:

```
(UGS, X, -100, 100, Y, 0, 80)
N199 (DIS."MILLING CUTTER D =16")
N200 S1500 T8.8 M6
N201 (RPT, 2)
N202 G X90 Y20 M3
N203 Z-100
N204 G1 Z-105 F150
N205 X40 F200
N206 G2 Y60 I40 J40
N207 G1 X90
N208 G Z-100
N209 (MIR, X)
N210 (ERP)
N211 (MIR)
N212 Z
```

ROT (URT) - Interpolation Plane Rotation

ROT is a part program command that rotates the active interpolation plane by a programmed angular value. The centre of rotation is the current origin. ROT can be activated through MDI or as a part program code.

Syntax

(ROT, *angle***)**

where:

angle	Represents the value of an angle expressed in decimal degrees. You can program the angle directly with a value numerical value or indirectly with an E parameter. Positive angles are measured CCW from the abscissa of the current interpolation plane. Negative angles are CW. If the angle is zero plane rotation is disabled.

Characteristics:

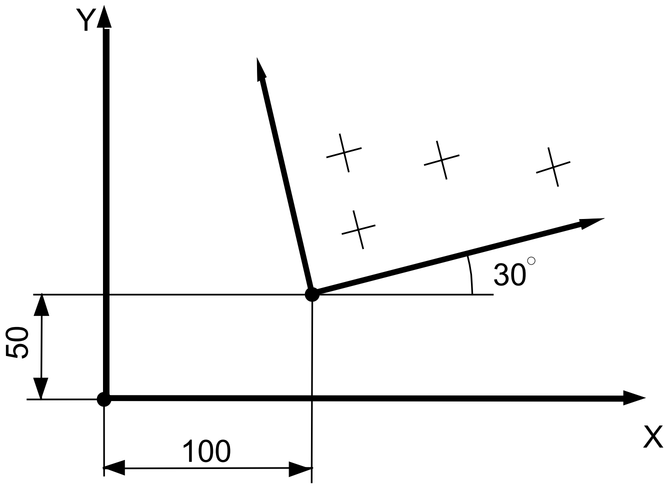
The control rotates programmed coordinates beginning with the first block after the ROT command. Coordinates referred to the machine zero (G79) are not rotated.

If you use axes rotation (ROT) in conjunction with mirroring (MIR), the control performs them in the following order: MIR first then ROT.

IMPORTANT

The system RESET disables the plane rotation. It is equivalent to programming (ROT,0).

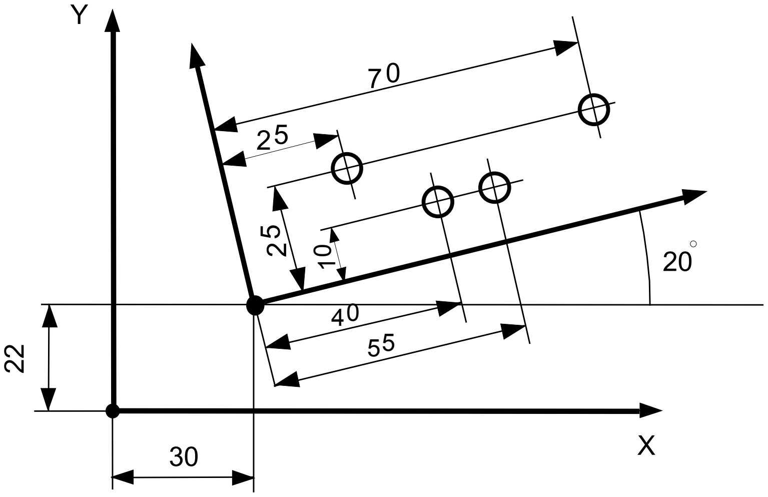
Example 1:



Program:

(UTO, 1, X100, Y50)	Activates absolute origin 1 with temporary origin (absolute offset) X100 and Y50
(ROT, 30)	Specifies 30 degree rotation CCW referred to temporary origin
.	
.	Moves in this portion of the program are referred to the temporary origin and rotated 30 degrees CCW
(UAO, 1)	Reactivates absolute origin 1
(ROT, 0)	Deactivates rotation by specifying a 0 degree rotation around origin 1

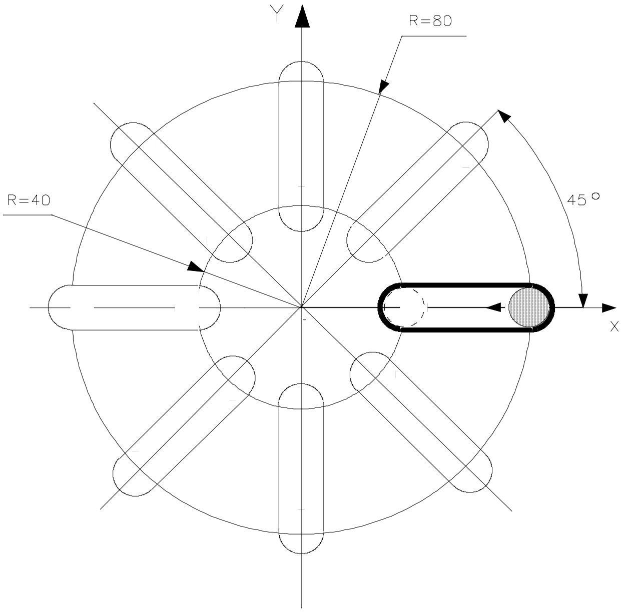
Example 2:



Program:

```
(UGS, X, 0, 70, Y, 0, 70)
N99 (DIS, "DRILLING D=6")
N100 S2000 F200 T3. 03 M6
N101 (UTO , 1, X30, Y22)
N102 (ROT, 20)
N103 G81 R-90 Z-110 M3
N104 X25 Y25
N105 X40 Y10
N106 X55
N107 X70 Y25
N108 G80 Z
N109 (UAO, 1)
N110 (ROT, 0)
N111 S1000 T4.4 M6
```

Example 3:
This example shows how to use Plane Rotation (ROT) with Repeat (RPT) and Parametric Programming.



Program:

```
(UGS, X, -100, 100, Y, -100, 100)
N148 (DIS, " ...")
N149 S1500 T5.5 M6
N150 E25 =0
N151 (RPT, 8)
N152 ( ROT, E25)
N153 G X40 Y M3
N154 Z0
N155 G29 G1 Z-10 F150
N156 X80 F200
N157 Z-18 F150
N158 X40
N159 G Z0
N160 E25 =E25 + 45
N161 (ERP)
N162 (ROT, 0)
```

UAO - Using Absolute Origins

The UAO commands lets you activate and use one of the absolute origins stored in memory.

Syntax

(UAO, *n* [,*axis1*, **. . . ,** *axis9*]**)**

where:

n	Defines the absolute origin number. It may be an integer between 0 and 10 or an E parameter. 0 enables the Home Position.
axis1,...,axis9	Is the address of the axis referred to absolute origin *n*.

Characteristics:

The UAO command allows up to nine axes. Only one absolute origin can be active at a time for a specific axis.

When an axis is not specified in the UAO command, it continues to use its currently active absolute origin. A UAO command programmed without axes (UAO, n) activates origin n for all axes. At power-up, after a control reset, or with n=0 and no axes, all axes are referred to the home position.

If the program requires different origins for different axes, program a separate UAO command for each origin required.

The origin values are automatically converted into and displayed in the unit of the current active mode (G70/G71).

The origins are referred to the HOME position that has been characterised in AMP.

Example:

(UAO,1)	Activates origin 1 for all axes.
.	
.	This portion of the program uses origin 1 for all axes.
.	
(UAO, 2, X, Y)	Activates absolute origin 2 for axes X and Y only.
(UAO, 3, Z)	Activates origin 3 for the Z axis.
.	
.	This portion of the program uses origin 2 for X and Y, origin 3 for Z,
.	and origin 1 for all other axes.
(UAO, 1)	Reactivates origin 1 for all axes.
.	
.	
.	
(UAO, 0)	Reactivates Home position for all axes.

UTO (UOT) - Using Temporary Origins

The UTO command temporarily increments the position of the specified absolute origin by a programmed amount for each declared axis.

Syntax

(UTO, *n* **,***axis1* [**,***axis2***, . . . ,** *axis9*]**)**

---

where:

n	Defines the number of the absolute origin to be temporarily modified. It is an integer from 0 to 10. If n=0, the programmed offset is added to the home position value.
axis1,...axis9	Is an axis and a dimension. The control treats the dimension as an absolute offset and adds it to the value of the absolute origin for that axis.

Characteristics:

You must declare at least one axis with a dimension in the UTO block but may declare up to nine axes with dimensions. An axis can only be declared once in each UTO command.

The axis dimension in the UTO command must be programmed in the currently active measuring unit (G70/ G71).

If an axis is omitted from in the UTO block, the current absolute origin for that axis stays active.

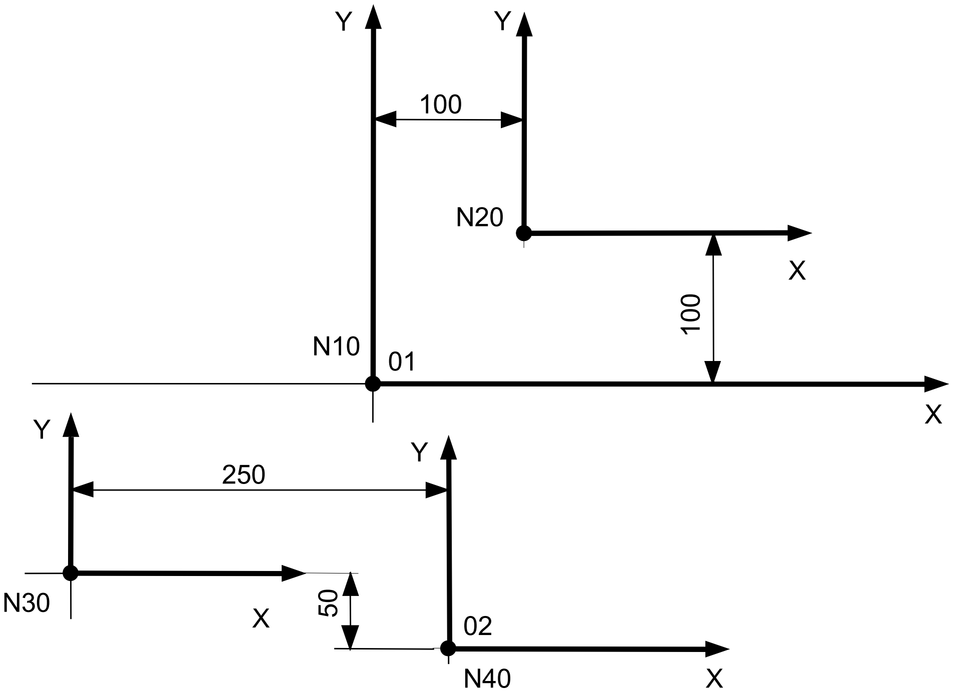
Once you activate a temporary origin it stays active until you:

- activate a new temporary origin with the UTO command
- activate a new absolute origin with the UAO command
- perform a control reset.

If a scale factor (SCF) is set, the control applies it to the UTO temporary origin.

Example 1:
Dimensions can be E parameters as shown in the block below:
(UTO,1,XE100)

Example 2:



Program:

N10 (UAO,1)	Activates absolute origin 1 for all axes.
.	
.	In this portion of the program all axes use origin 1.
.	
N20 (UTO, 1, X100, Y100)	Applies a temporary origin (absolute offset) to origin 1, X100 and Y100.
.	
.	In this portion of the program axes use origin 1, with X100 as temporary origin.
N30 (UTO, 2, X-250, Y50)	Activates a temporary origin (absolute offset) to origin 2, X-250 and Y50.
.	
.	This portion of the program uses absolute origin 2, with X- 250
.	and Y50 as temporary origin and origin 1 for the other axes.
N40 (UAO, 2)	Re-establishes absolute origin 2 for all axes.

UIO - Using Incremental Origins

The UIO command causes an incremental shift of the currently active origin for each axis specified in the command.

Syntax

(UIO, *axis1* [,*axis2*, **. . . ,** *axis9*]**)**

---

where:

axis1,...axis9	Is an axis and a dimension. The control treats the dimension as an incremental offset and adds it to the value of the current origin for that axis.

Characteristics:

You must declare at least one axis in the UIO command, but you may declare up to nine axes. An axis can only be declared once in each UIO command.

The axis dimension in the UIO command must be programmed with the current measuring unit (G70/ G71).

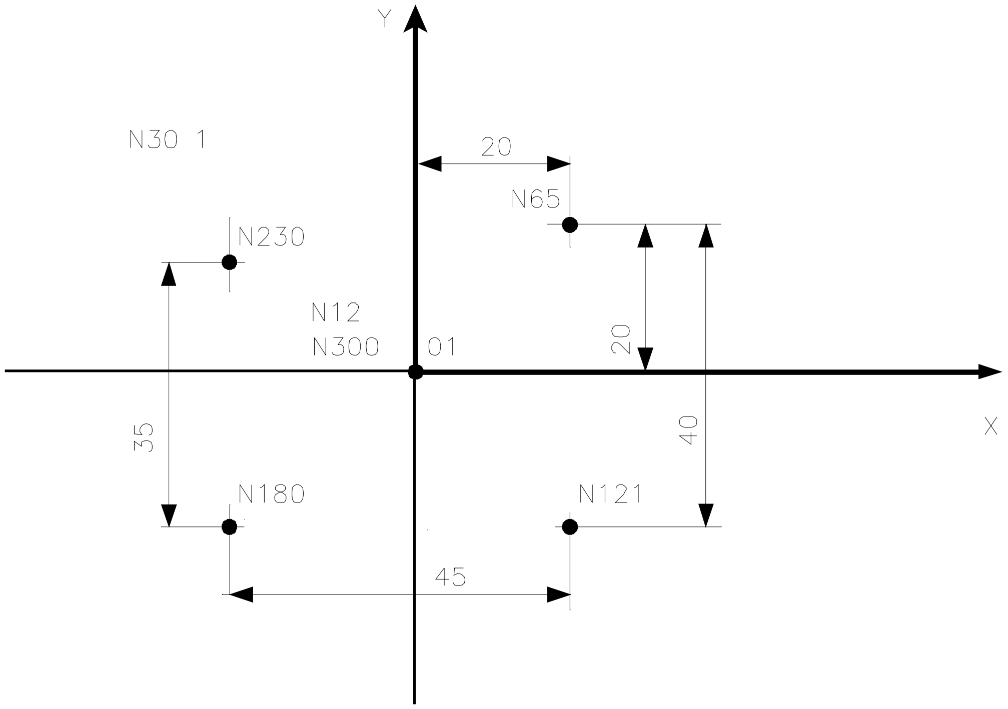
If an axis is omitted from in the UIO block, the current absolute origin for that axis stays active.

Once you activate an incremental origin for an axis it stays active until you:

- activate a new incremental origin for the axis with the UIO command
- activate an absolute origin with the UAO command
- perform a control reset.

If a scale factor (SCF) is set, the control applies it to a UIO incremental origin.

Example:



Program:

N12 (UAO,1)	Activates absolute origin 1 for all axes.
N65 (UIO, X20, Y20)	Applies an incremental offset of X20 and Y20 from origin 1. Absolute origin 0 for other axes remains in effect.
N121 (UIO, Y-40)	Applies a Y-40 increment to the last origin. The X20 incremental origin remains in effect.
N180 (UIO, X-45)	Applies an X-45 increment to the last origin. The Y-40 incremental origin remains in effect.
N230 (UIO, Y35)	Applies a Y35 increment to the last origin. The X-45 increment remains in effect.
N300 (UAO,1)	Restores absolute origin 1 for all axes.

RQO - Requalifying Origins

The RQO command lets you requalify, i.e. update and modify, an absolute origin from program. The RQO command modifies the specified origin by the specified amount for each axis you declare in the block. The origin must be stored in the origins table.

Syntax

(RQO, *n* **,***axis1* [**,***axis2***, . . . ,** *axis9*]**)**

where:

n	Defines the absolute origin number (1 to 10). The absolute origin must be stored via softkey driven procedure. n can be programmed directly with a positive integer number, or indirectly with an E parameter.
axis1,...axis9	Is an axis address and a dimension. The dimension is the increment added to the specified absolute origin of the specified axis and can be programmed directly with a decimal number, or indirectly with an E parameter.

Characteristics:

You must specify at least one axis and its dimension in the RQO command. Up to nine axes with their dimensions can be programmed. Program a specific axis only once for each RQO command.

The dimensions specified in the RQO command must be in the default measuring unit. No scale factor (SCF) must be applied to dimensions programmed in an RQO command.

The origin is requalified both in the origins file (so that the requalification becomes permanent) and in memory (if the origin is active when the requalification is applied).

In the table of the origins the requalification values are applied in the unit of measure in which the selected origin is expressed.

In the case of a diametrical axis, the requalification dimension must be programmed in radial terms, since the dimension is an increment to be added to the value already present in the origin, which is a radial value.

Example:

(RQO, 3, X (E31))	Modifies absolute origin no. 3 for axis X of the value contained in E31.

OVERTRAVELS AND PROTECTED AREAS

The commands in this class define the overtravel limits and the protected areas as described below.

COMMAND	FUNCTION
SOL	Defines software overtravels
DPA	Defines protected areas
PAE	Enables a protected area
PAD	Disables a protected area

SOL (DLO) - Software Overtravel Limits

The SOL command defines axis travel limits measured from the current origin.

Syntax

(SOL,*axis-name***,***lower-limit***,***upper-limit***)**

---

where:

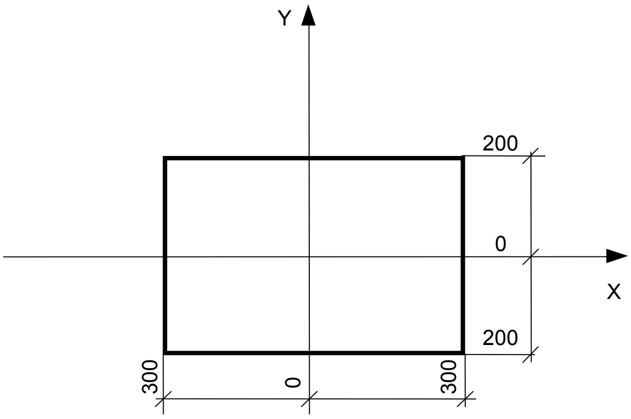
axis-name	Is the name of the axis whose travel limits must be defined.
lower-limit	Is a dimension for the lower limit.
upper-limit	Is a dimension for the upper limit. It must be greater than the lower limit.

Characteristics:

If the programmed software travel ends exceed the limits specified in AMP, the control signals an error.

Software overtravels must be programmed in the measuring unit (G70/G71) that is active when you program the SOL command. Active scale factors (SCF) are applied to the travel limits.

Example:



Program:

```
(SOL, X, -300, 300)
        .
        .
        .
(SOL, Y, -200, 200)
```

DPA (DSA) - Define Protected Areas

The DPA command defines a protected area.

Syntax

(DPA, *n*,*axis-name1*,*lower-limit1*,*upper-limit1*,*axis-name2*,*lower-limit2*,*upper-limit2* **)**

---

where:

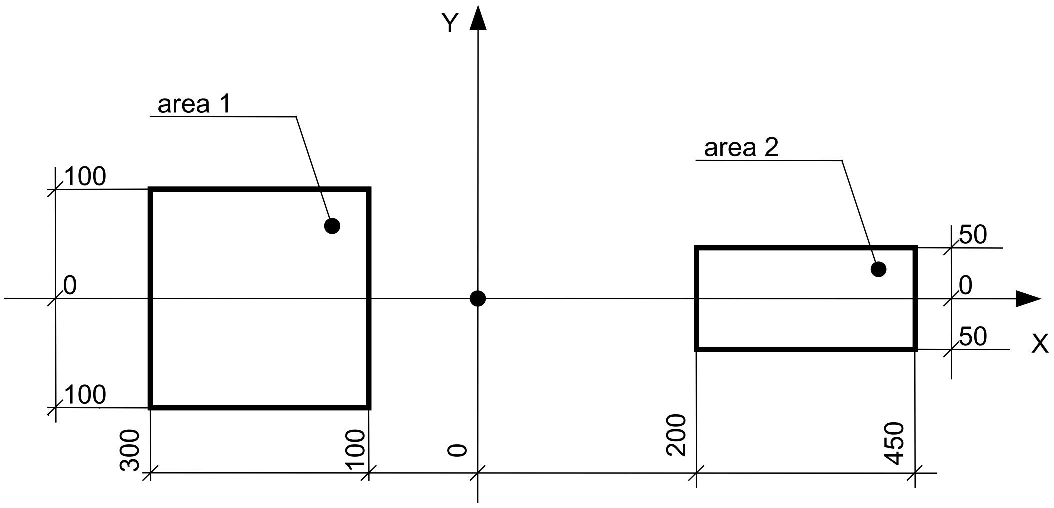
n	Is the area identifier number (1-3).
axis-name1	Is the name of the abscissa.
lower-limit1	Is the lower abscissa limit.
upper-limit1	Is the upper abscissa limit.
axis-name2	Is the name of the ordinate.
lower-limit2	Is the lower ordinate limit.
upper-limit2	Is the upper ordinate limit.

Characteristics:

Before beginning a programmed move with linear or circular interpolation, the control checks if the move enters a "protected area".

In a program you can define up to three protected areas for the machine tool. Each area is referred to the origin that is active when the DPA command is specified.

Example:



Program:

```
N1 (DPA,1, X, -300, -100, Y, -100, 100)
N2 ( DPA, 2, X, 200, 450, Y, -50, 50)
N3 (PAE, 1)
N4 (PAE, 2)
N5 T1.1 M6
    .
    .
    .
N80 (PAD, 1)
    .
    .
    .
N99 M30
```

PAE (ASC) - Protected Area Enable

The PAE command enables protected area control.

Syntax

(PAE, *n***)**

where:

n	Is the number of a protected area defined with the previous DPA command (1-3 range).

Example:
(PAE,2) enables area 2 control

PAD (DSC) - Protected Area Disable

The PAD command disables protected area control.

Syntax

(PAD, *n***)**

where:

n	Is the number of the protected area to be disabled (1-3 range).

Example:
(PAD,3) disables control on area 3.

VIRTUAL AXES MANAGEMENT

Virtual axes management provides a series of features that allow to manage the actual machine too axes in any three-dimensional space.

Virtual Axes

Virtual axes are declared in the configuration but are not associated to any physical parameter (for further information, refer to the AMP Manual).

Virtual axes are used:

- to handle planes rotated in space
- when cartesian planes are transformed into planes measured in polar coordinates
- when cartesian planes are transformed into planes measured in cylindric coordinates
- to perform moves in the tool direction
- to perform tool length compensation during the execution of profiles with five simultaneous axes, such as a tool mounted on a wrist with two degrees of freedom.

IMPORTANT

Prior to performing circular moves (G02-G03) on a programmed virtual plane you must define the interpolation plane with the G16 code.

When virtual axes are disabled the system forces the default plane configured in AMP as interpolation plane.

Virtual axes are disabled by a system reset.

Virtual modes available on 10 Series CNC

COMMAND	FUNCTION
UPR	(USE PLANE ROTATED) Rotation of cartesian axes XY and Z in space.
UVP	(USE VIRTUAL POLAR) Machining transformations from cartesian coordinates to polar coordinates
UVC	(USE VIRTUAL CYLINDRICAL) Machining transformations from cartesian coordinates to cylindrical coordinates.
TCP	(TOOL CENTER POINT) Programming in part ranges rather than in machine ranges.

UPR - Rotation of Cartesian axes

UPR (USE PLANE ROTATED) allows definition of a system of three virtual axes translated and rotated.

Syntax

(UPR,*type, af1af2af3, av1av2av3*[*rot1,rot2,rot3* [,*q1,q2,q3* [*,hor,ver*]]]**)**
(upr,*type, af1af2af3, av1av2av3*[*rot1,rot2,rot3* [,*q1,q2,q3* [*,hor,ver*]]]**)**
(UPR)
(UPR,99)
(UPR, 5, *af1af2af3, av1av2av3, type_req, pos-xx-H, pos-xx-V, n, ref-pos-xx-H, ref-pos-xx-V***)**

where:

type Specifies the rotation mode:

- 0= absolute
  rotation angles are referenced to physical axes
- 1= incremental
  rotation angles are referenced to the system of axes used for incremental programming.
  This type of rotation is possible if a previous rotation has been programmed (absolute the first time, incremental for subsequent times.)
- 2= absolute, with transformation on 5 axes
  The characteristics are similar to those of type 0 but the rotation of the linear axes influences the position of the two rotating axes. Upon activation of the UPR function, the rotating axes take on a new value which identifies the position of the tool axes with respect to the virtual system.
- 3= incremental, with transformation on 5 axes
  The characteristics are similar to those of type 1 and also encompasses the points outlined in the previous item.

- 4 = rotating plane automatically determined according to the direction of the tool axis.

- 5 = does not define a new virtualisation: it enables you to request the physical or virtual positions of the rotary axes relative to the currently active virtualisation.

- 6 = as 4 but with 5-axis transformation (see 2).

- 10 = global, type 0
  The characteristics are similar to those of type 0, but the programmed rotation remains active even with subsequent non global UPR’s.

- 12 = global, type 2
  The characteristics are similar to those of type 2, but the programmed rotation remains active even with subsequent non global UPR’s.

- 14 = global, type 4
  The characteristics are similar to those of type 4, but the programmed rotation remains active even with subsequent non global UPR’s.

- 16 = global, type 6
  The characteristics are similar to those of type 6, but the programmed rotation remains active even with subsequent non global UPR’s.

- 99
  without any further parameter, disables all active UPR modes, including global rototranslation.

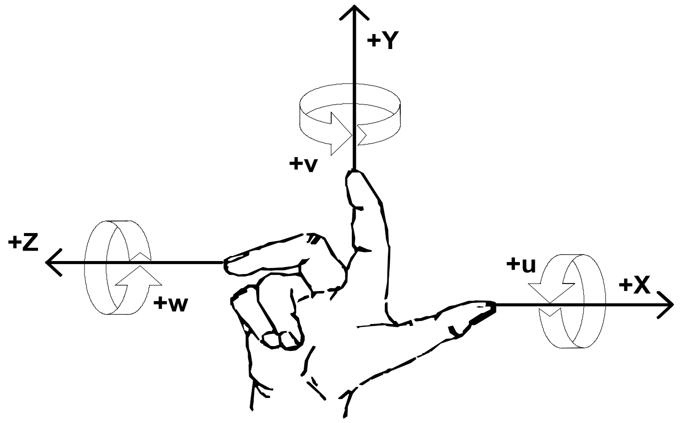
af1af2af3	Names of the three physical axes to be handled in virtual mode (e. g. XYZ).
av1av2av3	Names of the three virtual axes to be moved (e. g. UVW).
rot1,rot2,rot3	Rotation angles expressed in degrees. Direction is selected with the right hand rule, which is discussed in the next section.
q1,q2,q3	These parameters allow the origin of the reference system to determine the origin of the reference system: • In an absolute rotation, q1, q2 and q3 are the co-ordinates of the system origin. They are relative to the origin that was active when UPR was programmed. • In an incremental rotation, q1, q2 and q3 are the increments applied to the current origin. The new system origin is the result of adding these increments to the co-ordinates of the current one. If *q1,q2,q3* are not present, the origin of the reference system coincides with the origin of the physical system being used.
hor,ver	Values in degrees of angles interior to the round angle (between -360° and +360°) identifying the origin of the horizontal and vertical rotary axes. This must always be regarded as translations of the physical axes.
Type_req	Type of position requested of the system: • **0** = **physical** position • **1** = **virtual** position
pos-xx-H	Angle expressed in degrees; this is the physical or virtual position of the horizontal rotary axis to which the rotation matrices must be applied in order to determine the corresponding position requested.
pos-xx-V	Angle expressed in degrees; this is the physical or virtual position of the vertical rotary axis to which the rotation matrices must be applied in order to determine the corresponding position requested.

n	**E** variable **indicator** to be used as a starting point to save the values of the requested angles as returned by the system: the first pair (horizontal axis position, vertical axis position) corresponds to the solution closest to the ref-pos-xx angles, the second pair (Hor, Ver) is complementary to the first.
ref-pos-xx-H	Angle expressed in degrees; it is the value around which the solution for the horizontal axis must be determined.
ref-pos-xx-V	Angle expressed in degrees; it is the value around which the solution for the vertical axis must be determined.
no parameters	(UPR) without parameters disables the UPR mode, leaving the global mode only, if present. If no global rotation has been programmed, commands (UPR) and (UPR,99) are equivalent.
lower case programming (UPR)	If the three-letter block is programmed in lower case, the angles and/or origins can be disabled using the UPR algorithm without exiting from the "CONTINUOUS MOVEMENT" mode. The lower case three-letter block (upr,...... must only be used to alter the parameters of the (UPR, ...) programmed in advance. The axis sequence cannot be altered. A global type rotation cannot be programmed.

WARNING

In a 5-axis transformation, the behaviour of the lower case upr may, in some instances, differ from that of the upper case UPR.
Since in continuous mode synchronisation with the physical positions is not applicable, the values of the angles determined by the system have been normalised relative to 0.0 and therefore might turn out to be complementary to the desired values. At any rate, the solutions provided for the tool position retain their validity.

To select the direction of rotation of the system of virtual axes you must apply the right hand rule.



Right hand rule

Characteristics:

UPR allows programming of any machine function in a space that has been rotated by the specified angle with respect to the machine tool Cartesian system.

This lets you program the profile in the normal Cartesian space (XYZ) and then have 10 Series recalculate the axes moves according to the virtual planes resulting from rotation.

By programming a **global UPR** you can apply an initial rototranslation to serve as a term of reference for all subsequent UPR's. This makes it possible to perform the same part program, which, in its turn, may contain other geometric transformation instructions, even when the part to be processed is, for instance, fastened to the bench by brackets set along different directions.

The activation of a global type UPR is not rated as a previous UPR for purposes of incremental or lowercase programming.

The activation of a global type UPR cancels any previous virtualisation, resulting in the reset of any displacements.

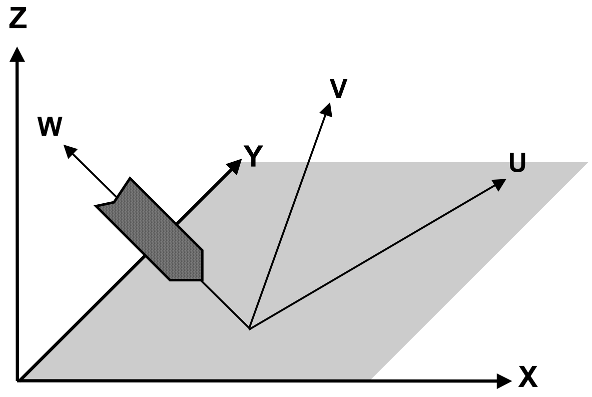
When a **UPR type 2, 3, 4 or 6** is used, or any of the corresponding global type UPR's are used, the system needs the machine characteristics, which must be entered in the **TCP** table according to the rules established by the TCP itself. This feature is separate form TCP, or rather, it can work whether TCP is active or not. When UPR and (TCP, 5) are active simultaneously, the movement of the axes will influence the movement of the virtual axes.

With **modes 2 and 3**, the tool axis, once identified by the programming of the rotating axes, takes on the same position within the virtual reference system as it has in the reference system identified by the Cartesian system.

Modes 4 and 6 and the corresponding global type modes determine the new rotation plane based on tool direction (i.e. based on the positions of the rotary axes). On the new plane, supposing the programming UPR, 4, XYZ, UVW ..., the W axis will coincide with the direction of the tool, the U axis will lie on the original XY plane and, consequently, the V axis will be based on the rule that determines the Cartesian axes.

A type **UAO or UTO** origin **cannot** be programmed with an active virtualisation.

Origins on rotary axes implemented with a **UIO** command are not managed.



By means of angles *rot1, rot2 and rot3*, the *UVW* axes determined in this way may be further rotated; an increase (see type 1 or 3) will be applied to the new axes.

The **RESET** command disables the UPR mode.

IMPORTANT

When a **global UPR** is programmed, the values of *rot1, rot2, rot3* must be between –90° and +90°.

For angles outside this range, when passing from one virtualisation to another, the algorithm generating the rotary axis position could generate unwanted rotations (unwanted, but still congruent with the position of the tool).

IMPORTANT

The **UPR is insensitive to the DAN**: the axes to be specified in the UPR are the names of the physical and virtual axes desired, regardless of any DAN commands executed previously.

IMPORTANT

Starting with release 7.4 in order to remedy operating anomalies and solve solution indeterminacy problems during a virtualisation involving a 5-axis transformation, the algorithms employed to determine the positions of rotary axes have been changed. Downward compatibility is ensured by defining the **!R73MODE** user variable.

!R73MODE = 1 : the calculation of the virtualisation is by means of the old algorithms, in line with release 7.3
!R73MODE = 0 or not defined : the calculation of the virtualisation is by means of the new algorithms as are used in release 7.4 or higher

This user variable is read solely in conjunction with upper case UPR programming.

A programming error is notified if the rotary axes origins are programmed in the three-letter UPR code and one of the following conditions occurs:

- user variable !R73MODE is set on 1. This is so because the afore-mentioned programming has been managed starting with release 7.5 and conflicts with the use of the calculation algorithms associated with release 7.3.
- the type of UPR programmed does not require a 5-axis transformation.

Rotation modes:

The three-axis cartesian system rotates sequentially by the programmed angles. This means that:

A) The *af1 af2 af3* system of coordinates rotates by a *rot1* angle around the *af1* axis

B) The new *av1' av2' av3'* system of coordinates, which results from the above described rotation, rotates by a *rot2* angle around the *av2'* axis

C) The *av1" av2" av3"* system of coordinates, which results from the rotations described in A) and B), rotates by a *rot3* angle around the *av3'* axis

After these three operations have been completed, the virtual axes of the resulting system will be *av1 av2 av3.*

IMPORTANT

Since steps A) B) and C) are carried out sequentially, the order in which angles and axes are programmed in the block is critical.

Using UPR

In the following examples it is supposed that the the three-axis cartesian system XYZ is the reference system.

In the examples 1, 2, 3, 4, 5 the reference system origin coincides with the origin in use at the time of programming UPR.

Example 1:

```
.
.
GXYZ
(UPR,0,XYZ,UVW,30,45,60)
G1F5400
U100.4V9.12 W-70
.
U70.345 W-20
.
.
.
.
.
G16UV
G02 U100 V70 R15
G1 U120 W10
.
.
(UPR)
X1 Z4.9
.
```

The UVW system may be obtained by:

A) rotating by 30 degrees the XYZ system around the X axis

B) rotating the U'V'W' system resulting from A) by 45 degrees around V'

C) rotating the U''V''W'' resulting from A) and B) by 60 degrees around W''

Example 2:

```
.
.
.
GXYZ20
(UPR,0,XYZ,UVW,10,0,80)
G1F4000
U50 V70
.
U90
V80 W60
.
.
(UPR)
.
.
.
```

The UVW system may be obtained by:

A) rotating the XYZ system by 10 degrees around the X axis.

B) rotating the U'V'W' system resulting from A) by 80 degrees around the W' axis

Example 3:

```
.
.
.
GXYZ20
(UPR,0,ZYX,ABC,80,0,10)
G1F3000
A50 B70
.
A90
B80 C60
.
.
(UPR)
.
.
```

The ABC system may be obtained by:

A) rotating the XYZ system by 80 degrees around the Z axis

B) rotating the A'B'C' system resulting from A) by 10 degrees around the C' axis

IMPORTANT

The UVW virtual system discussed in Example 2) is different from the ABC system in Example 3).

Example 4:

```
.
.
.
(UPR,0,ZYX,WVU,0,50,60)
U90V30
W10
.
.
.
.
(UPR)
.
.
.
```

The WVU system may be obtained by:

A) rotating the XYZ system by 50 degrees around the Y axis.

B) rotating the W'V'U' system resulting from A) by 60 degrees around the U' axis

Example 5:

```
.
.
GXYZ
(UPR,0,XYZ,UVW,30,0,0)

G16 UV
G1 F1000 U50 V0
V30
.
.
U25 V35
(UPR,1,XYZ,UVW,60,0,0)
U30 V20
.
.
(UPR)
GX10 Y25
.
.
```

The UVW system may be obtained by rotating the XYZ system by 30 degrees around the X axis

The UVW system resulting from the previous rotation rotates by another 60 degrees around U

Example 6:

```
.
GXYZ
(UPR,0,XYZ,UVW,30,45,60,10.8,20,-30.2)
G1 F5400
U100.4V9.12W-70
U70.345 W-20
.
.
.
G16UV
G02 U100 V70 R15
G1 U120 W10
.
.
(UPR)
X1 Z4.9
.
```

The UVW system may be obtained by:

A) rotating the XYZ system by 30 degrees around the X axis.

B) rotating the U'V'W' system resulting from A) by 45 degrees around V

C) rotating the U''V''W'' resulting from A) and B) by 60 degrees around W''

The origin of the reference system coincides with the point whose coordinates are X10.8, Y20, Z-30.2

Example 7:

```
.
.
GXYZ
(UPR,0,XYZ,UVW,30,45,60,10.8,20,-30.2)
G1F500
U100.4 V9.12 W-70
U70.345 W-20
.
.
.
.
.
```

The UVW system may be obtained by:

A) rotating the XYZ system by 30 degrees around the X axis.

B) rotating the U'V'W' system resulting from A) by 45 degrees around V'.

C) rotating the U''V''W'' resulting from A) and B) by 60 degrees around W''

The origin of the reference system coincides with the point whose coordinates are X10.8, Y20, Z-30.2

```
.
(UPR,1,XYZ,UVW,10,0,0,3,8,5)
U120 V30
.
.
.
(UPR)
GX70.5 Y10 Z25
.
.
```

The UVW system resulting from C) rotates by another 10 degrees around U.

The origin of the reference system coincides with the point whose coordinates are X13.8, Y28, Z-25.2

Example 8:

In this case, we want to apply an initial rotation, say of 90° around the Z axis, to the part program discussed in example 2:

```
.
.
GXYZ
(UPR,10,XYZ,UVW,0,0,90)
(UPR,0,XYZ,UVW,10,0,80)
G1F4000
U50 V70
U90
V80 W60
.
.
(UPR,99)
```

The UVW system is achieved through:

A) a 90° rotation of the XYZ system around the Z axis

B) a 10° rotation of the U'V'W' system originating from point A) around U'

C) an 80° rotation of the U"V"W" system originating from point B) around W"

Disables all active UPR's

Example 9:

Utilisation of type 5 upr.

Let us assume we are applying to a Double Twist type machine tool a virtualisation involving a 90° rotation around the Z axis.

For the sake of brevity, let us denote with B the horizontal rotation axis, with A the vertical rotation axis, with WRK the corresponding virtual positions and with ABS the corresponding physical positions.

	Positions read on screen	
	WRK	ABS
XYZAB	A 0 B 0	A 0 B 0
(UPR,2,XYZ,UVW,0,0,90)	A 0 B -90	A 0 B 0

If you want the system to determine the physical positions corresponding to virtual positions A=0 and B=0, with the solutions in the E variables being entered starting from indicator 10, program the following upr:

	Positions read in the E variables:
(UPR,5,XYZ,UVW,0,0,0,10,0,0)	E10 = 90, E11 = 0, E12 = -90, E13 = 180

UVP - Programming polar coordinates

UVP permits to program axes moves with virtual coordinates.

Syntax

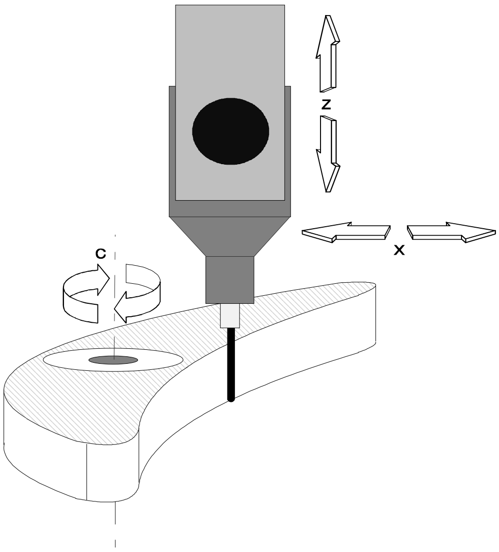
(UVP,*af1af2*,*av1av2*,r**)**
(UVP)

where:

af1	physical linear axis (for example, X)
af2	physical rotary axis (for example, C)
av1	virtual abscissa axis (for example, u)
av2	virtual ordinate axis (for example, v)
r	minimum radius the tool path must not enter
no parameters	disable UVP by programming (UVP) without parameters

Characteristics:

This virtual mode permits to move an X linear axis and a C rotary axis by programming their coordinates in a UV cartesian plan. The (U,V) coordinates of any point on the virtual plane are translated into the (X,C) coordinates of the physical axis.



Machining with polar coordinates

IMPORTANT

To avoid the rotary axis is requested to exceed the rapid feed, the ***r*** parameter must be programmed bear in mind the F programmed feed.

The minimum radius should be calculated using the following formula:

$$r = \frac{F}{Vmax} * \frac{360}{2\pi}$$

where:

r = minimum radius

F = programmed feed (mm/min or inch/min)

Vmax = rotary axis rapid speed

Whether to the **r** parameter is attributed a negative value, the feed limitation is executed dinamically, in such a way to allow high feeds in working point that are quite far from the working centre.

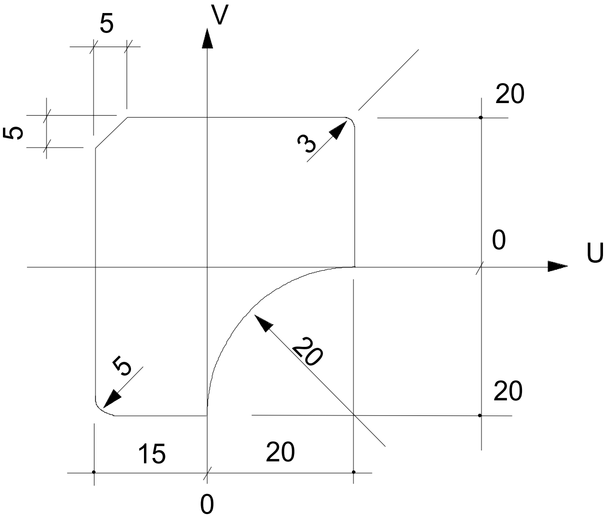
IMPORTANT

Accelerations are calculated in a dynamic mode, therefore it is preferable to use this (**r-)** programming only when positioned at a certain distance from the working centre as, in this case, the rotary axis might be subject to too high accelerations

Programming examples with polar coordinates

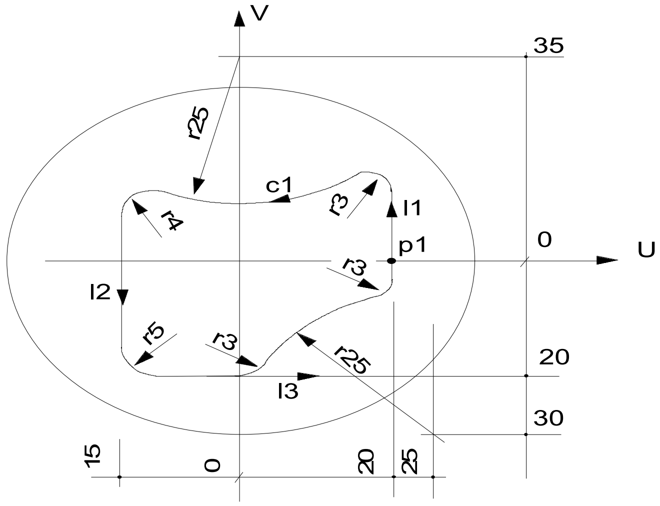
Example 1:

```
E0=110*180/(3.14159*800)
T1.1M6
S1000M3
GC0X50Y0
(UVP,XC,UV,E0)
G16 UV
G1G42U20VF110
V20
r3
U-15
b5
V-20
r5
U0
G40G2U20V0I20J-20
(UVP)
GX50
```



Example 2:

```
T1.1M6
S2000M3F300
GC0X80Y0-Z-5
(UVP,XC,UV,10)
G16UV
p1=U20V0
l1=p1,a90
c1=I0J35r-25
l2=U-15V0,a90
l3=U0V-20,a0
c2=I25J-30r-25
G21G42p1
l1
r3
c1
r4
l2
r5
l3
r3
c2
r3
l1
p1
G20G40
(UVP)
GX80
```



UVC - Programming cylindrical coordinates

The UVC command (USE VIRTUAL CYLINDRICAL) permits to program cylindrical coordinates.

Syntax

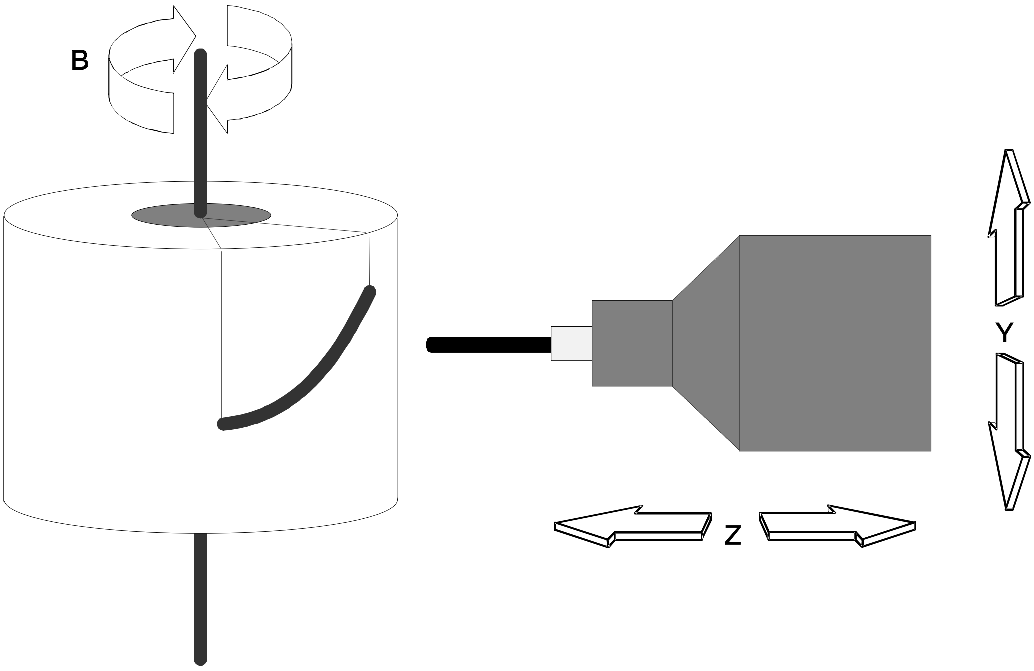
(UVC,_af1,av1,_r**)**
(UVC)

where:

af1	physical rotary axis (for example, B)
av1	virtual axis (for example, W)
r	cylindrical radius on which the profile is executed
no parameters	without parameters disables **UVC** mode.

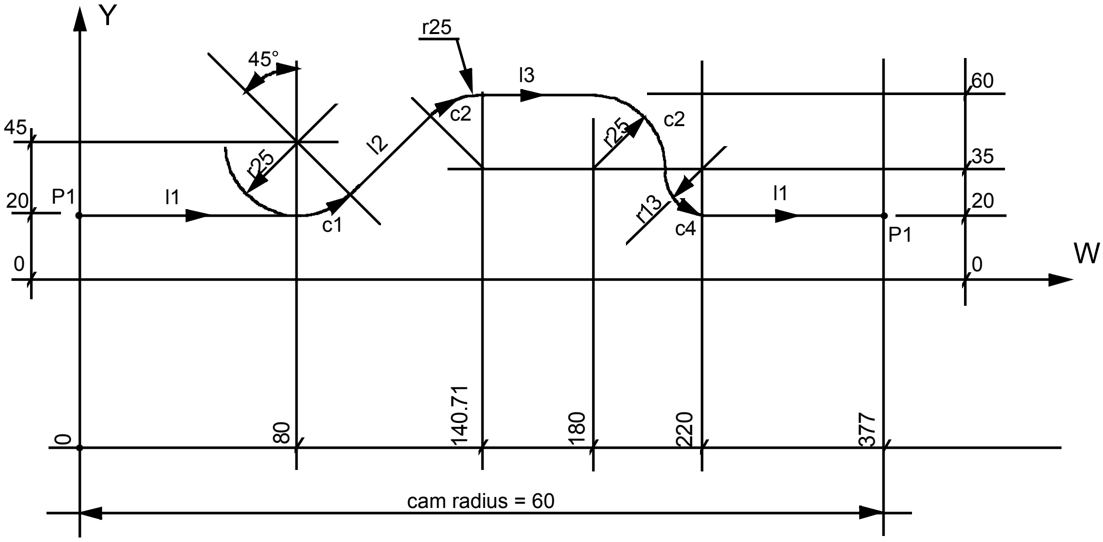
Characteristics:

This mode permits to move a Y linear axis and a B rotary axis by programming their coordinates on a WY plane. While the moves of virtual Y coincide with those of physical Y, each move of W corresponds to a circle arc, which is a function of the cylinder radius and must be translated into an angular move of the B physical axis.



Machining with cylindrical coordinates

Example:



```
(DIS,"EXAMPLE UVC");- Programming with cylindrical coordinates -
T1.1M6
S2000F300M3
B0
XY20Z10
E30=60;CYLINDER RADIUS
(UVC,B,W,E30)
G16WY
p1=W0Y20
E31=2*3.1415*E30
p2=WE31Y20
l1=p1,p2
c1=I80J45r25
c2=I140.71J35r-25
l2=c1,c2
c3=I180J35r-25
l3=c2,c3
c4=c3,l1,r15
G21G41p1
Z-12
l1
c1
l2
c2
l3
c3
c4
l1
G20G40p2
GZ20
(UVC)
M30
```

UVA - Programming non orthogonal axes

The UVA (USE VIRTUAL ANGULAR) command makes it possible to program non orthogonal axes by treating them as if they were orthogonal.

Syntax

(UVA,*af1af2*,*av1av2*,a**)**
(UVA)

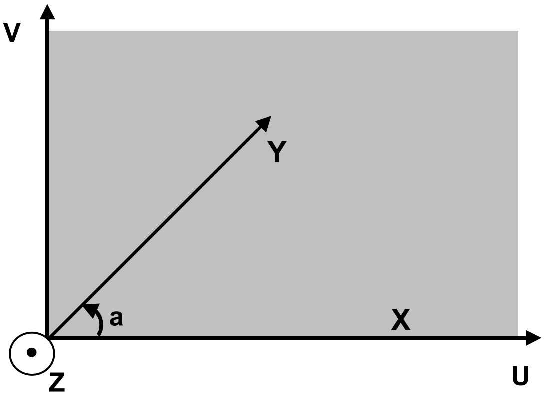
---

where:

af1	is the name of the first physical linear axis (e.g. X)
af2	is the name of the second physical linear axis (e.g. Y)
av1	is the name of the virtual abscissa axis (e.g. U)
av2	is the name of the virtual ordinate axis (e.g. V)
a	is the angle, in degrees, subtended by two positive semi-axes of the physical linear axes
no parameters	**(UVA)** without parameters disables the UVA mode.

Characteristics:

This type of virtualisation is used when you want to move two linear X Y axes that are not mutually orthogonal by programming the coordinates on a virtual orthogonal UV plane. The U axis, the abscissa of the new pair of axes, will be coaxial with axis X, whereas axis V will be perpendicular to U, forming an angle of 90°- with axis Y.
The origin of UV will coincide with the origin of XY. The position of a generic point P with coordinates (U,V) in the virtual plane is translated into the (X,Y) coordinates of the physical axes. While the movements of U coincide with those of the corresponding physical axis (X), every movement of V is translated into an interpolated movement of X and Y.



View of the XY plane and the corresponding UV plane

IMPORTANT

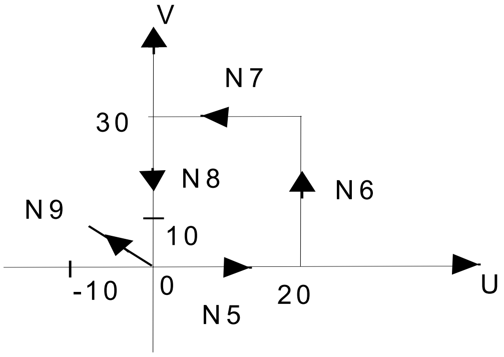
The sign of parameter ***a*** is positive if the X axis, as it reaches its position on the Y axis, rotates around itself in the counterclockwise direction; it is negative if the direction of rotation is clockwise (in the example shown in the figure below, ***a*** is positive).

Values of ***a*** of 0° and 180° **are not permissible**, as this would entail two linear physical axes arranged parallel to one another.

Example of programming with polar coordinates

Example 1:

```
N1 T1.1M6
N2 S1000M3
N3 GX0Y0
N4 (UVA,XY,UV,45)
N5 G1U20VF110
N6 V30
N7 U0
N8 V0
N9 U-10V10
N10 (UVA)
N11 GX50
```



TCP - Tool Center Point for machines with "Double Twist" head

This feature permits to refer the programs of a 5-axis machine (with 3 linear + 2 rotary axes) to the tool tip rather than to the center of rotation of the axes (or the head center).

The position controlled by the system depends on position of the rotary axes and on the geometric characteristics of the head.
The algorithm may be activated by simply programming the TCP three letter code in a block;

it is disabled by programming (TCP) or by giving a RESET command.

If the three-letter code is programmed in lower case (tcp,n), some of the parameters can be altered without disabling the TCP algorithm and without exiting from"CONTINUOUS MOVEMENT" mode. The lower case three-letter code must only be used to alter the parameters of the (TCP,n) programmed previously, and the *n* parameter must be the same as specified earlier; furthermore, (tcp,n) must not be used in mode n=5.

Syntax

(TCP[,n])
(tcp,[n])

---

where:

n	Offset mode code (1÷5); the various modes are shown in the pages that follow.

Characteristics:

In the TCP block you must also write information about the current tools. The system may handle a head simultaneously mounting as many as four different tools.

Configuration parameters must be assigned after the head has been positioned at the angles shown in figure 2.1. These parameters are included in the user table. There are three tools you can use for entering data in the user table: the user table editor, the specific machine logic functions, or the L variables in the program. Also you may upload the predefined configuration files available with the SETUP utility of the USER TABLE EDITOR.

NOTE:

- For more information about user table management from PLUS refer to the description of $TBLPUTD and $TBLGETD functions in the PLUS Library Manual.
- To access the user table from the TABLE EDITOR environment refer to the User Manual.

TCP Table for machines with "Double Twist" head

Variable	User Table	Record in PLUS User Table	Part Program	Meaning	TCP
Offset mode code	385	Record 97 - Var 1	L384	Defines the offset calculation mode according to the machine configura-tion. Allowed values are as follows: 0 - Enables dynamic part of TCP only, geometric calculation excluded 1 - No mobile table in the machine 2 - Mobile table on first axis 3 - Mobile table on second axis 4 - Mobile table on first and second axes 5 - Reserved.	1, 2, 3, 5
Tool length ①	386	Record 97 - Var 2	L385	Defines the length of the current mounted tool. The length is obviously referred to the active tool holder.	1, 2, 5
Active tool holder [1]	387	Record 97 - Var 3	L386	Active tool holder number 1÷4.	1, 2, 3, 5
Parameter A ① (figures 2.2 e 2.3)	388	Record 97 - Var 4	L387	Specifies the distance --with sign and expressed in mm-- between the rotation axis of the horizontal head and the parallel plane supporting the rotation axis of the vertical head.	1, 2, 5
Horizontal head offset①	389	Record 98 - Var 1	L388	It is the offset --expressed in degrees-- to be applied to the horizontal head axis so that the plane that supports the tool coincides with the YZ plane of the machine. To calculate this offset value, rotate the horizontal axis until the plane that supports the tool coincides with the YZ plane of the machine, and then direct the vertical head towards Y- with a positive head move. Then pick up the displayed coordinate, multiply it by the rotation coefficient of the horizontal head (Horizontal head rotation coefficient) and change the sign of the result.	1, 2, 3, 4, 5
Horizontal head rotation ① (figure 2.3)	390	Record 98 - Var 2	L389	Defines horizontal head rotation as seen from above. Allowed values for positive programming are as follows: +1 for clockwise rotation - 1 for counter clockwise rotation.	1, 2, 3, 4, 5
Vertical head rotation① (figure 2.2)	391	Record 98 - Var 3	L390	Defines vertical head rotation as seen from the left hand side. Allowed values for positive programming are as follows: +1 for clockwise rotation - 1 for counter clockwise rotation.	1, 2, 3, 4, 5

① Parameter alterable in continuous through lower case programming (tcp,n).

TCP Table (continued)

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
1st linear axis ID	392	Record 98 - Var 4	L391	ID of the axis to be used as abscissa in the offset algorithm.	1, 2, 3, 4, 5
2nd linear axis ID	393	Record 99 - Var 1	L392	ID of the axis to be used as ordinate in the offset algorithm.	1, 2, 3, 4, 5
3rd linear axis ID	394	Record 99 - Var 2	L393	ID of the axis to be used as vertical axis in the offset algorithm.	1, 2, 3, 5
Horizontal head ID	395	Record 99 - Var 3	L394	ID of the axis on which the horizontal head is mounted. If this ID=0, the head inclination angle will be the one specified in the *Horizontal Head Inclination* parameter.	1, 2, 3, 5
Vertical head ID	396	Record 99 - Var 4	L395	ID of the axis on which the vertical head is mounted. If this ID=0, the head inclination angle will be the one specified in the *Vertical Head Inclination* parameter.	1, 2, 3, 5
Horizontal head inclination	397	Record 100 Var 1	L396	Horizontal head inclination angle to be used when the horizontal head axis ID=0. In this case the horizontal head is to the specified position. It is a value with sign that is a function of the configured "horizontal head direction".	1, 2, 3, 5
Vertical head inclination	398	Record 100 Var 2	L397	Vertical head inclination angle to be used when the vertical head axis ID=0. In this case the vertical head is to the specified position. It is a value with sign that is a function of the configured "vertical head direction".	1, 2, 3, 5
Radius ① (figure 2.1)	399	Record 100 Var 3	L398	Defines the radius of the tool currently mounted on the machine, with reference to the active tool holder.	1 2 5
Angle ① (figure 2.1)	400	Record 100 Var 4	L399	Defines the angle formed between the tool center and the tool contact point after the vertical head has been rotated in order for the tool to be in the Y- direction.	1 2 3 5

① Parameter alterable in continuous through lower case programming (tcp,n).

Tool holder 1

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B ① (figures 2.1, 2.3)	369	Record 93 - Var 1	L368	Distance with sign expressed in mm from the tool plane --in the Y- direction-- to the parallel plane that contains the horizontal head rotation axis.	1 2 5
Parameter C ① (figures 2.1, 2.2)	370	Record 93 - Var 2	L369	Distance with sign expressed in mm from the tool plane and the parallel plane that contains the vertical head rotation axis.	1 2 5
Parameter D ① (figures 2.2, 2.3)	371	Record 93 - Var 3	L370	Distance with sign expressed in mm from the vertical head rotation axis to the tool holder grip.	1 2 5
Offset vertical head ①	372	Record 93 - Var 4	L371	It is the offset, expressed in degrees, to be applied to the vertical head axis in order to position the tool axis parallel to the machine -Y direction. To calculate this value, position the tool in the Y- direction, read the displayed position, multiply it by the configured vertical head direction and change the sign of the result.	1 2 3 5

① Parameter alterable in continuous through lower case programming (tcp,n).

Tool holder 2

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B ① (figures 2.1, 2.3)	373	Record 94 - Var 1	L372	As in tool holder 1	1 2 5
Parameter C ① (figures 2.1, 2.2)	374	Record 94 - Var 2	L373	As in tool holder 1	1 2 5
Parameter D ① (figures 2.2, 2.3)	375	Record 94 - Var 3	L374	As in tool holder 1	1 2 5
Vertical head offset ①	376	Record 94 - Var 4	L375	As in tool holder 1	1 2 3 5

Tool holder 3

Variable	User Table	Record in PLUS user table	Part Program.	Meaning	TCP
Parameter B ① (figures 2.1, 2.3)	377	Record 95 - Var 1	L376	As in tool holder 1	1 2 5
Parameter C ① (figures 2.1, 2.2)	378	Record 95 - Var 2	L377	As in tool holder 1	1 2 5
Parameter D ① (figures 2.2, 2.3)	379	Record 95 - Var 3	L378	As in tool holder 1	1 2 5
Offset vertical head ①	380	Record 95 - Var 4	L379	As in tool holder 1	1 2 3 5

Tool holder 4

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B ① (figures 2.1, 2.3)	381	Record 96 - Var 1	L380	As in tool holder 1	1 2 5
Parameter C ① (figures 2.1, 2.2)	382	Record 96 - Var 2	L381	As in tool holder 1	1 2 5
Parameter D ① (figures 2.2, 2.3)	383	Record 96 - Var 3	L382	As in tool holder 1	1 2 5
Offset vertical head ①	384	Record 96 - Var 4	L383	As in tool holder 1	1 2 3 5

① Parameter alterable in continuous through lower case programming (tcp,n).

The figures that follow illustrate the offset characterization parameters applicable to a machine. The machine is seen from above, from the front and from the side.

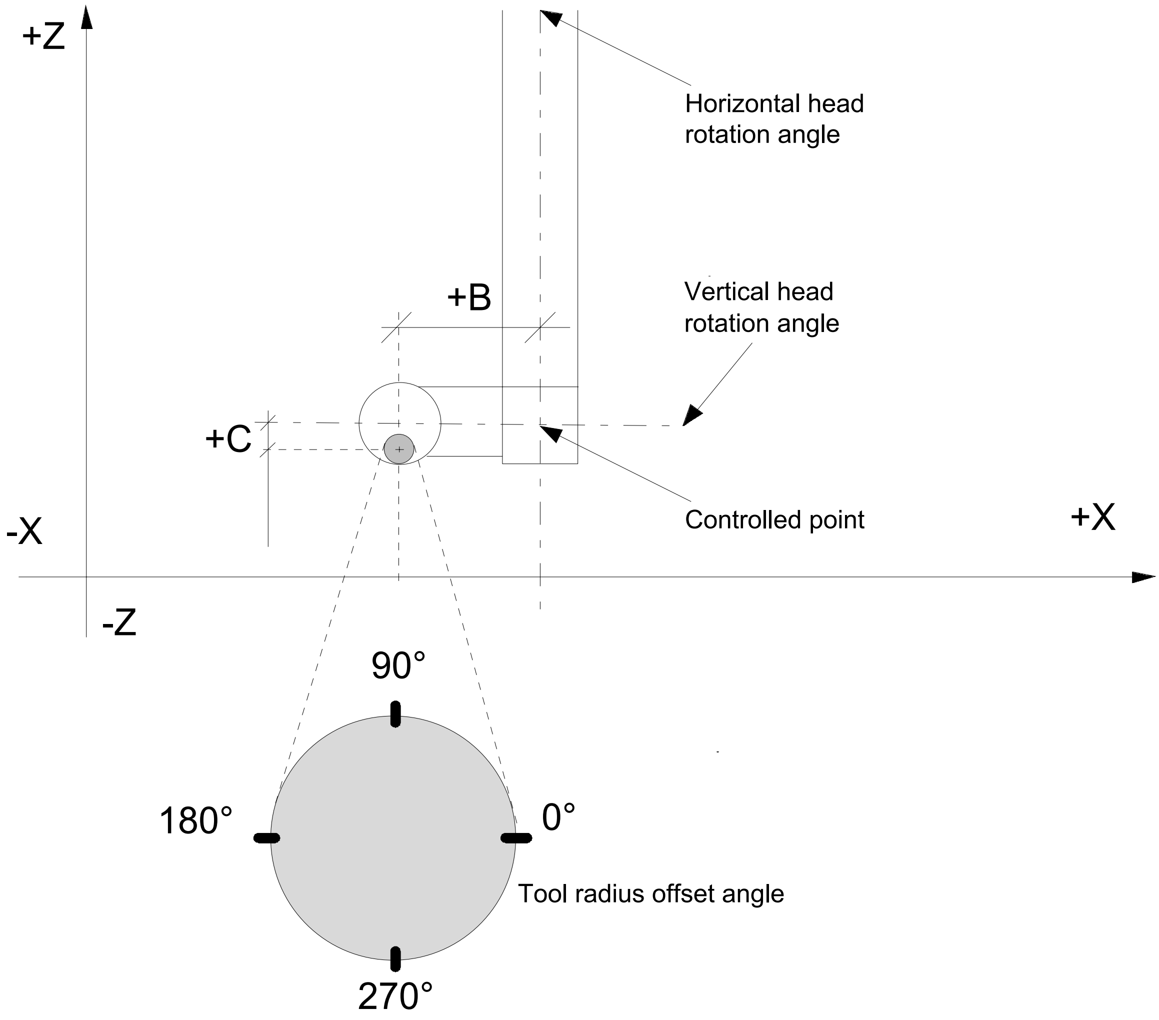


Fig. 2.1 Front View

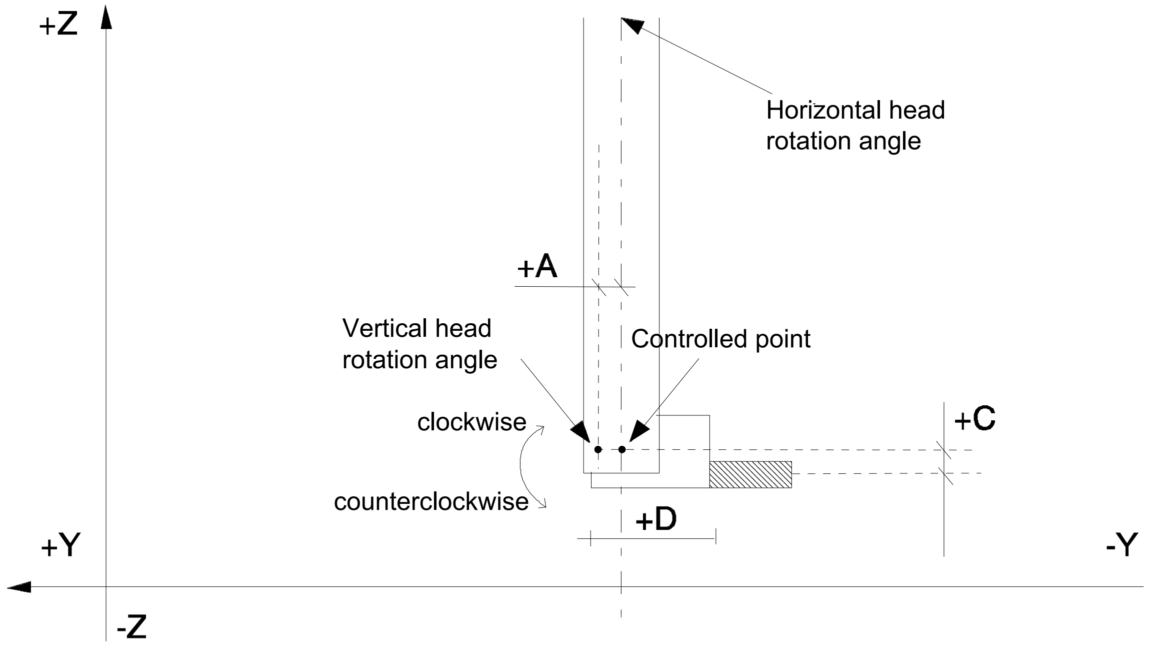


Fig 2.2 Side View

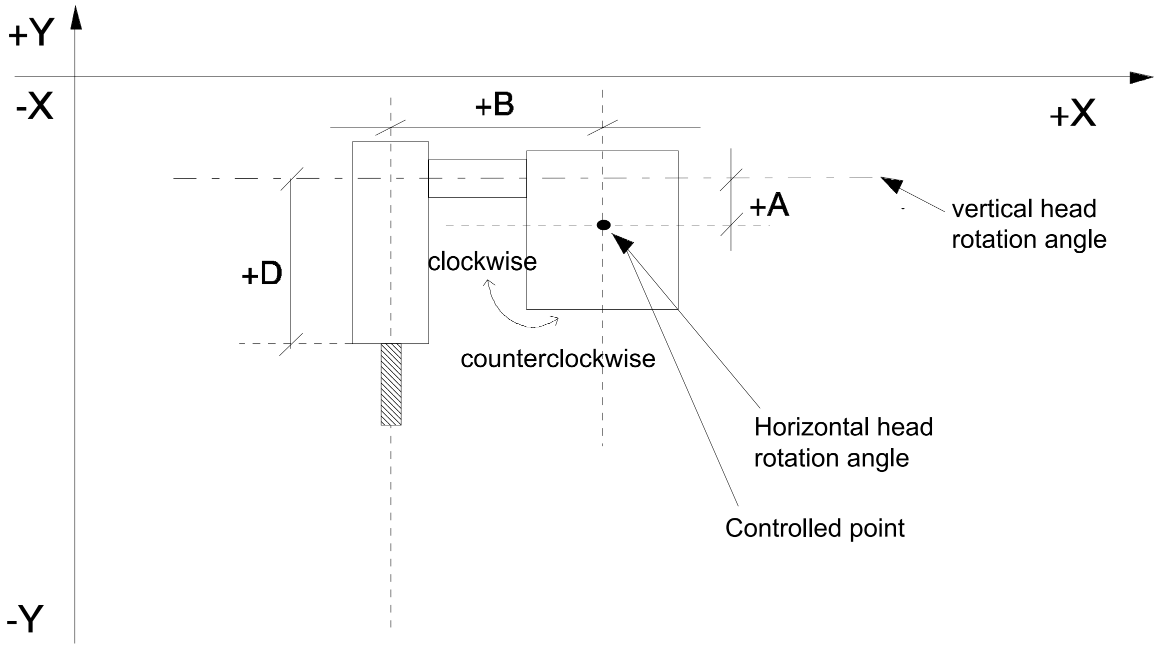


Fig 2.3 Machine seen from above

Dynamics

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Dynamic mode	357	Record 90 - Var 1	L356	These fields may have the following values: 0 - Speeds and accelerations of linear axes are not limited, and may exceed the configured values *run time* 1 - Speeds of linear and rotary axes are limited *run time*, if necessary, in such as never to exceed the configured values. Accelerations of the linear and rotary axes are limited in advance so that overlapping of movements does not require accelerations in excess of those configured. 2 - Speeds and accelerations of the linear and rotary axes are limited in advance so that the speeds and accelerations calculated *run time* never exceed the configured values. In this case, if for example only the linear axis is moved, it remains limited unlike what happened in case 1. It must be stressed that in case 2, the constant speed is guaranteed on the profile, which does not happen in case 1 if speed of an axis is cut *run time.*	1 2 3 5
Corner radius① (figures 2.5 and 2.6)	358	Record 90 - Var 2	L357	Radius of the tool corner in cases of spherical or toroidal tools	1, 2, 3, 5
Angle of contact a (figures 2.5 and 2.6)①	359	Record 90 - Var 3	L358	Angle a of contact for spherical or toroidal tools (0 £ to £90)	1, 2, 3, 5
Programming of "m", "n" and "o"	360	Record 90 - Var 4	L359	Defines how contact between tool and piece is to be handled. With values "0" and "1", the control is defined through angles which are in turn defined as follows: 0 = through U.T. variables 359 and 400 1 = through m and n from the part program With the 2 value, on the other hand, the contact is established through a normal vector on the profile defined in the 3 components m, n, o.	1, 2, 3, 5

① Parameter alterable in continuous through lower case programming (tcp,n).

Dynamics (continued)

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Threshold angle (figure 2.4)	361	Record 91 - Var 1	L360	Angle beyond which the system adds a horizontal rotary axis move --after the element has been completed-- in order to position the tool perpendicular to the subsequent element. Below this angle, the horizontal rotary axis move starts when the element is completed and continues during execution of the subsequent element.	2
Rotary axis velocity	362	Record 91 - Var 2	L361	Velocity expressed in degrees per minute at which the horizontal rotary axis moves when it is automatically positioned between two part program blocks. If it is 0 the system uses the programmed feedrate.	2
Interpolation mode	363	Record 91 - Var 3	L362	Specifies whether or not the rotary axes are interpolated in conjunction with the linear axes. If not interpolated, the rotary axes follow the linear axes moves. Allowed values are: 0 - Rotary axes interpolated in conjunction with linear axes 1 - Rotary axes follow linear axes Select mode 1 if the velocity on the profile should not be affected by rotary axes motion.	1 2 3 5
Interpolation type	364	Record 91 - Var 4	L363	Specifies whether the rotary axes must be interpolated only by the error or must be treated as linear axes. Allowed values are: 0 - Rotary axes interpolated as linear axes 1 - Rotary axes interpolated only by the error.	1 2 3 5
Integrator	365	Record 92 - Var 1	L364	With interpolation mode 1, this parameter defines whether acceleration/deceleration ramps will be used for rotary axes or whether velocities proportional to the linear axes moves will be used. Allowed values are as follows: 0 - No ramps 1 - Ramps If 1 is selected, the velocity diagram will be calculated using the configured parameters for rotary axes, irrespective of linear axes moves.	1 2 3 5

Dynamics (continued)

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Offset mode	366	Record 92 - Var 2	L365	Defines the modalities of activation of the length corrector and the tool radius (and possibly the tool corner radius based on the TTR variable). 0 - values taken from the TCP table 1 - values with sign active in the system (length 1 and tool radius) - absolute values active in the system (length 1 and tool radius) - absolute value of length 1 only; the tool radius is considered = 0.	1 2 3 5
Minimum move	367	Record 92 - Var 3	L366	Specifies the minimum angle beyond which the system automatically generates a move of the horizontal rotary axis in order to position the tool perpendicular to the subsequent element.	2
Travel limits control	368	Record 92 - Var 4	L367	Defines how the configured travel limits will be controlled. Allowed values are: 0 - travel limits controlled both in the program block (before execution) and at each point in real time. 1 - travel limits controlled both only at each point in real time. **NOTE:** Since part programs are typically referred to the tool tip, program block control is also referred to the tool tip. Real time control is referred to the center of rotation of the axes. When the system detects that the point is beyond the configured limit, it gives an error signal and locks the axes motion with deceleration. If the program block goes beyond the limit, instead, the system signals an error and the movement is not performed.	1 2 3 4 5

NOTE:

In the above tables distances are expressed in mm. They must be intended in the configured length unit.

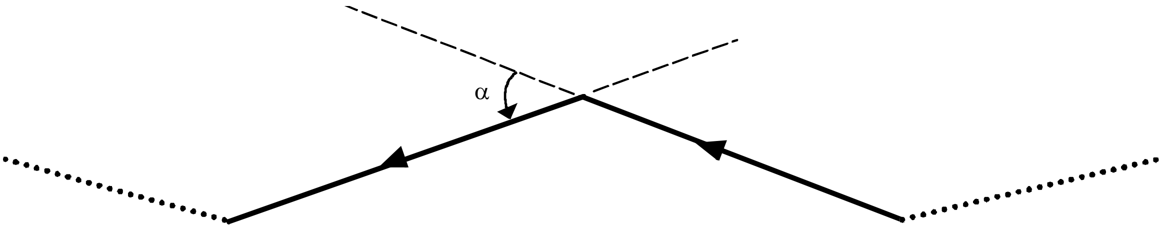


Fig. 2.4 Threshold angle (α)

The following figures illustrate significance of the TCP parameters used for management of spherical and toroidal tools; the abbreviation U.T. indicates the corresponding number in the User Table.

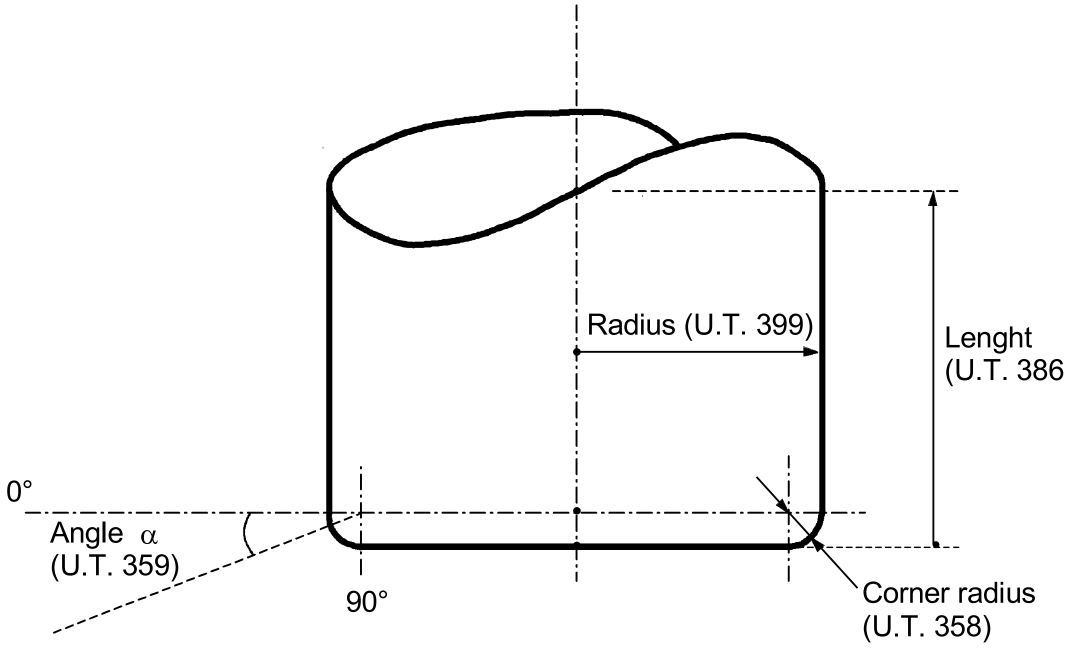


Fig. 2.5 Toroidal tool

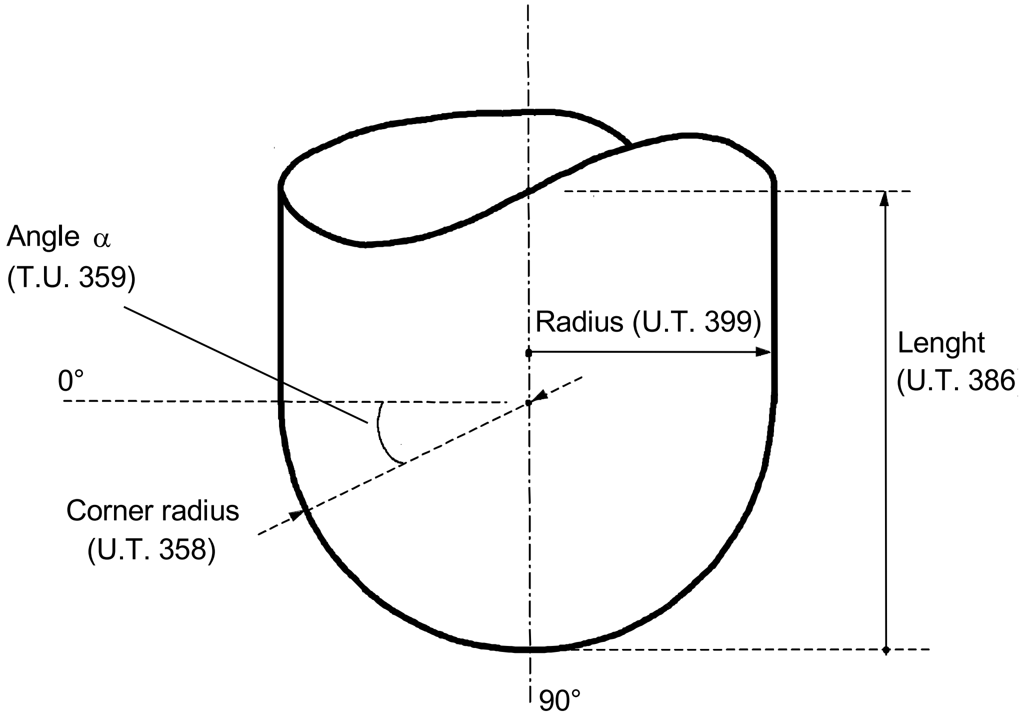


Fig. 2.6 Spherical tool

- **Mode 1 (TCP,1)**

With (TCP,1) programmed coordinates are referred to the part. This mode is recommended when the profile must be executed with both linear (1÷3) and rotary (1÷2) axes, the tool is expected to remain permanently in contact with the machining surface, and the linear axes coordinates are referred to the part.

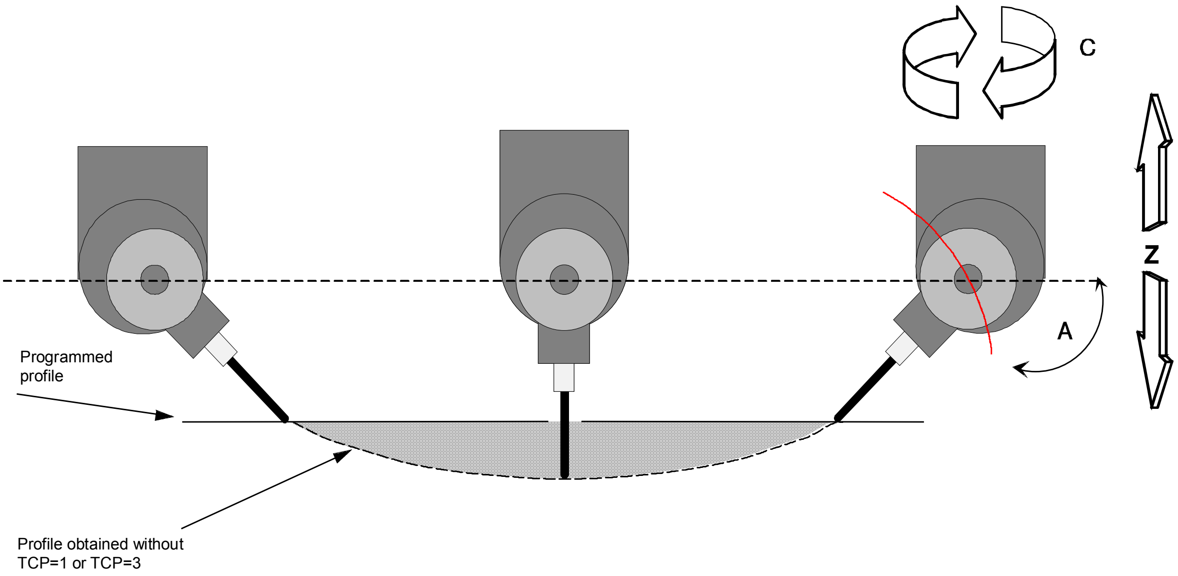
To keep the tool tip on the profile defined by the linear axes, the system offsets displacements by automatically adjusting the position of the three linear axes to the rotary axes.

- **Mode 3 (TCP,1)**

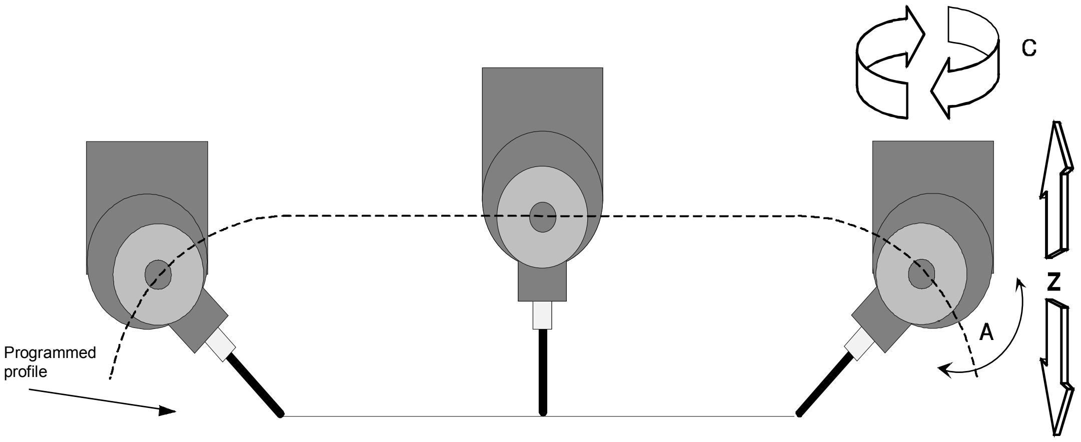
This mode permits to use CAD programs in which the tool length, radius, corner radius and contact angles are different from the ones with which the program (machine coordinates) was generated.

Tool length and diameter variances must be input via machine logic (refer to the $TCPWRT instruction in the PLUS Library Manual).

As with (TCP,1) in order to keep the tool tip on the profile defined by the linear axes, the system offsets displacements due to variations of length, radii and contact angles by automatically adjusting the position of the three linear axes to the rotary axes.



Profile obtained without TCP 1 or TCP 3



Profile obtained with TCP 1 or TCP 3

- **Mode 2 (TCP,2)**

Mode 2 is an extension of mode 1.

In addition to (TCP,1) features, Mode 2 incorporates a move of the horizontal rotary axis so that its inclination remains constant throughout the profile defined by the first two linear axes configured in the TCP table.

The horizontal rotary axis move may be generated:

- *between two elements*, to allow for rotation on bevels. In this case, the horizontal rotary axis moves either between the end of an element and the start of the subsequent one, or while the subsequent element is executed (refer to the Threshold Angle parameter).
- *on a circular element*, to keep inclination constant throughout the element. In this case the element and the horizontal rotary axis move will occur simultaneously.

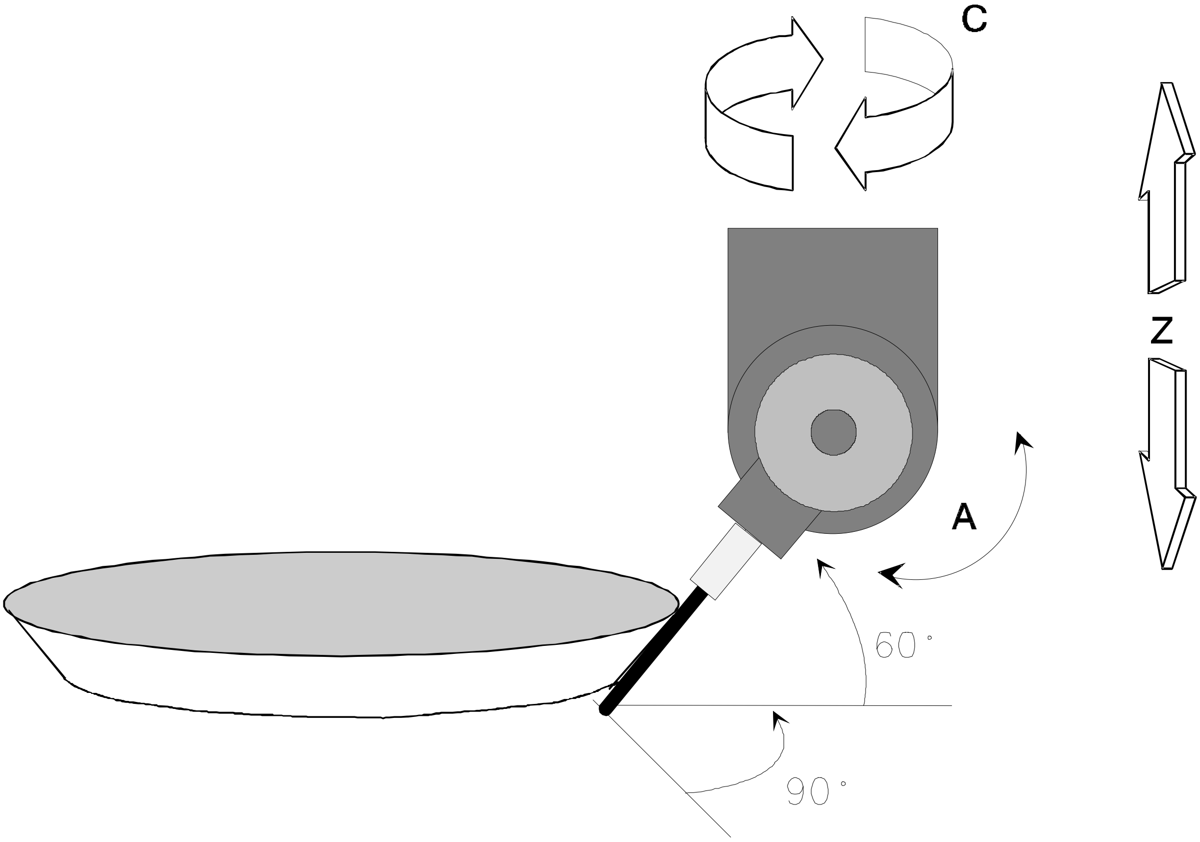
These rules apply to all the elements in the profile except for the first one. In fact, the first element is seen as the profile start point even when it is circular and therefore involves no rotary axis move. The system will start generating the horizontal rotary axis move from the second element.

- **Mode 4 (TCP,4)**

(TCP,4) permits the inclination of the horizontal rotary axis to remain constant throughout the profile defined by the first two linear axes configured in the TCP table.

The horizontal rotary axis is adjusted continuously, i.e. between each pair of interpolated points rather than between two elements as in (TCP,2).

For this reason the profile must not be discontinuous. If it is, a "SERVO ERROR" may occur.



Axes moves with TCP,2 or TCP,4

IMPORTANT

(TCP,4) does NOT include the features of (TCP,1).

- **Mode 5 (TCP,5)**

This mode permits to execute moves in the tool direction.

When (TCP,5) is active the system generates a virtual axis whose name has been configured in AMP (VIRTUAL AXES section of the process configuration).

The virtual axis move generates a move of the linear axes in which the tool displaces in its direction, i.e. in the direc(TCP,5) may be programmed both separately or in conjunction with the other axes, in which case (TCP,5) features add to (TCP,1) features. may be moved manually.

Mode 5 may be used for drilling inclined holes. Also, it may be used for removing the tool from the part when a failure interrupts the machining cycle. When this oxccurs, we may have three different cases:

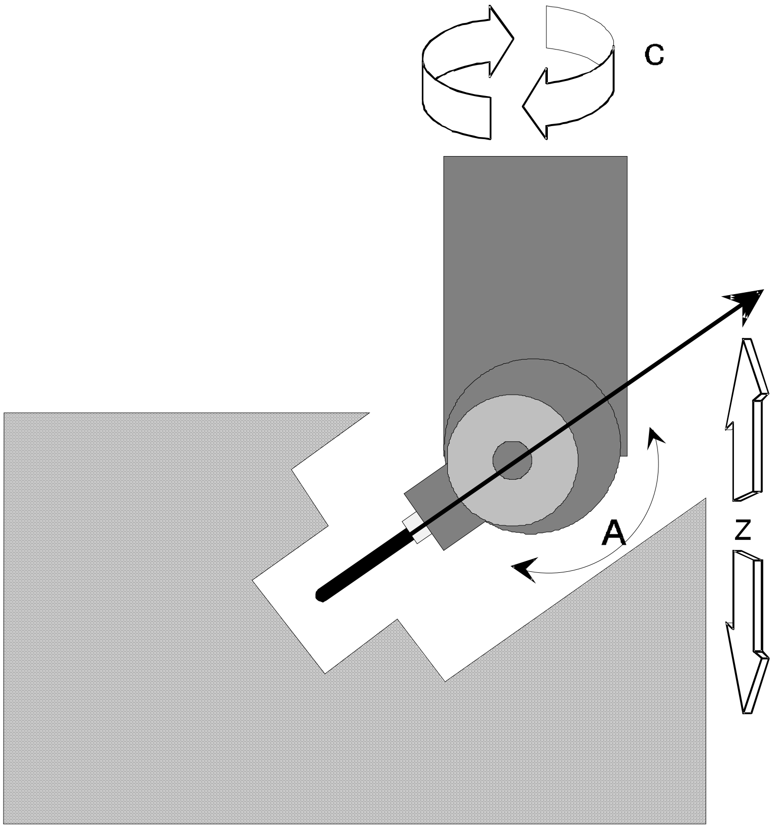
A) **(TCP,5) is enabled, the rotary axes are not homed, and the interruption occurred during a cycle using any TCP mode.**
In this case the system automatically selects the tool direction according to the position of the rotary axes before the failure. The same may occur after power off, because the rotary axes position is stored in the permanent memory. The values that determine the rotary axes position may be altered via machine logic by means of the MANUAL SETUP data entry or the $TCPWRT function. In particular, $TCPWRT permits to alter the direction in which the tool will be extracted.

B) **(TCP,5) is enabled, the rotary axes are not homed, and the interruption occurred during a cycle not using TCP.**
In this case the system *must* be given the tool direction, i.e. the position of the rotary axes, because the stored values may not be significant. The rotary axes position may be input via machine logic by means of the MANUAL SETUP data entry or the $TCPWRT function.

C) **(TCP,5) is enabled, the rotary axes are homed.**
In this case the tool direction is given by the current rotary axes position rather than by their position when the interrupt occurred, no matter whether a TCP mode was active during the previous move. However, in normal working conditions stored and current rotary axes positions coincide.

NOTE:

For more information about the MANUAL SETUP data entry or the $TCPWRT function please refer to the PLUS Library Manual and the User Manual.



Removing the tool with TCP,5

NOTE:

In all TCP modes, the machine logic may send to the interpolator in real time any variance of tool length and tool radius, or of the corner radius, as well of the angles formed between the tool center and the contact point (parameters 386, 399, 400, 358 and 359 in the User Table).

This permits to offset tool wear during machining (refer to the description of the $TCPWRT function in the PLUS Library Manual).

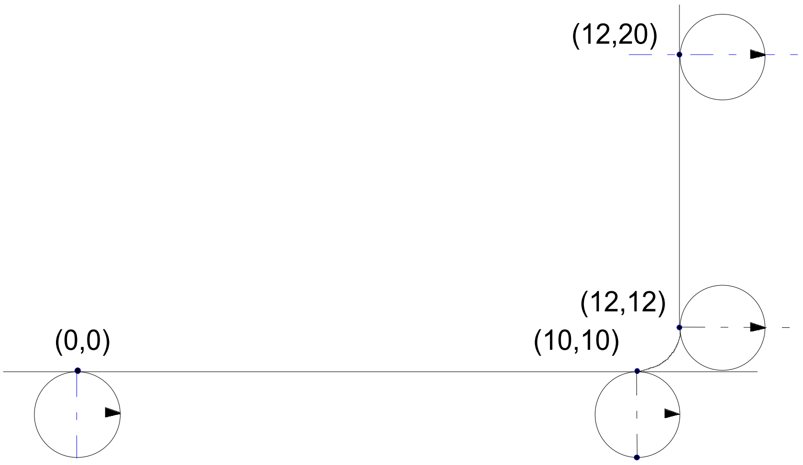
Programming the "m" and "n" parameters (angles)

In all the modes of use of TCP described above, in addition to the axis positions, it is also possible to program the "m" and "n" parameters, significance of which is as follows:

m	angle subtended between the tool center and the point of contact of the tool with the piece (coincides with U.T. parameter 400) - see figures 2.1, 2.5 and 2.6.
n	angle of contact between cutting corner and piece (coincides with parameter 359) see figures 2.5 and 2.6.

By programming these parameters (enabled by setting U.T. variable 359 to 1), you can vary the point of contact between tool and piece from one block to the next.

Example:



```
G1X10Y0m270
(TCP,1)
G1XY
X10Y10
G3X12Y12I10J12m180
G1X12Y20
(TCP)
```

Programming the "m", "n", "o" and “u”, “v”, “w” parameters (vectors)

In all the TCP utilisation modes described above, in addition to the axis positions it is possible to program the "m", "n", "o" which stand for the 3 components of the vector normal to the section to be machined, and the “u”, “v”, “w” parameters denoting the paraxial compensation factors to which the tool radius compensation must be applied.

By programming these parameters it is possible to change the piece-tool contact point from one block to another and through this vector you can obtain the tool diameter compensation.

To enable tool radius compensation by means of the “m”, “n”, “o” parameters, set variables T.U. 360 and T.U. 366 on 2. These parameters are applied to the 1st, 2nd, 3rd axis configured in the TCP table.

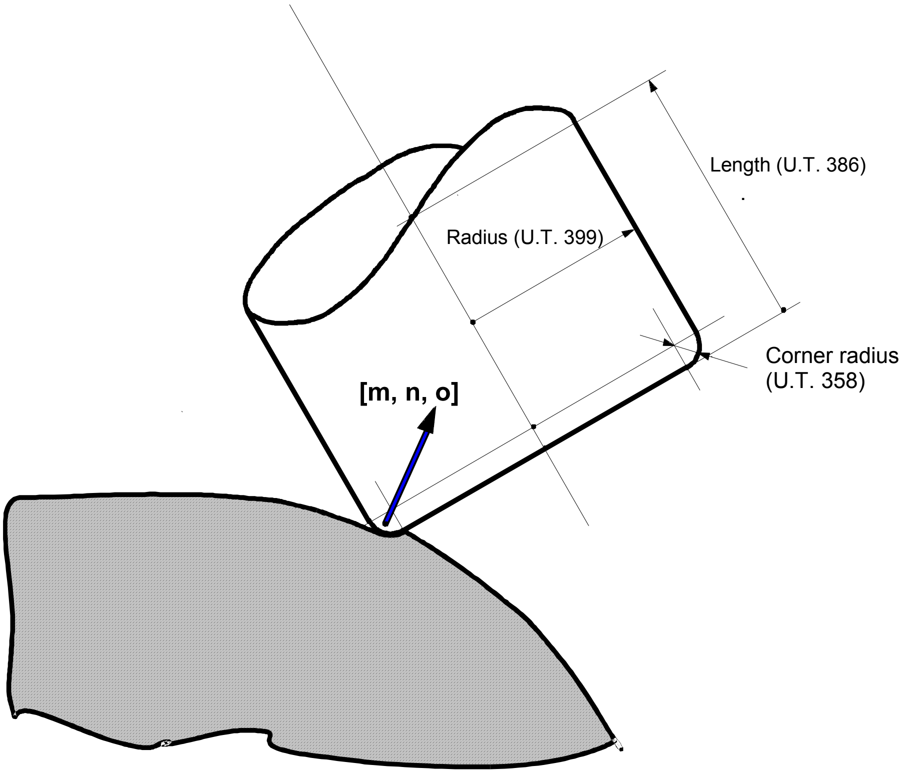
To enable tool radius compensation by means of the “u”, “v”, “w” parameters set the T.U. 360 variable on 0 and the T.U. 366 variable on 3. These parameters are applied to the 1st, 2nd, 3rd axis configured in the process.

Example of programming with “m”, “n”, “o”:

```
m..n..o..          ;initial value of vector
h1
(TCP,....)         ;TCP activation
X..Y..Z.. AB m..n..o..
X..Y..Z.. AB m..n..o..
X..Z.. m..n..o..
A..B.. m..n..o..
X..Y..
.
.
.
.
.
(TCP)
```

Example of programming with “u”, “v”, “w”

```

h1
(TCP,....)         ;TCP activation
X..Y..Z.. AB u..v..w..
X..Y..Z.. AB u..v..w..
X..Z..u..v..w..
A..B..u..v..w..
X..Y..
.
.
.
.
.
(TCP)
```



TCP - Tool Center Point for generic 5-axis machines

This feature permits programming of a 5-axis machine (with 3 linear + 2 rotary axes) irrespective of tool orientation and type (spherical or toroidal). The position controlled by the system is that of the point of attachment of the tool in the spindle (see figures 2.7 and 2.8).

The algorithm may be activated simply by programming the TCP code in a block; it is disabled by programming (**TCP**) or by giving a RESET command.

Syntax

(TCP[,n])

where:

n Offset mode code (1, 3, 5); the various modes are illustrated in the pages that follow.

Characteristics:

In the TCP block you must also write information about the current tools: these tools must be inserted in the "User Table" - abbreviated U.T.

There are three tools you can use for entering data in the user table: the user table editor, the specific machine logic functions, or the L variables in the program.

Also you may upload the predefined configuration files available with the SETUP utility of the USER TABLE EDITOR.

NOTES:

- For more information about user table management from PLUS refer to the description of $TBLPUTD and $TBLGETD functions in the PLUS Library Manual.
- To access the user table from the TABLE EDITOR environment refer to the User Manual.

The following figures illustrate significance of the TCP parameters used for management of spherical and toroidal tools; the abbreviation U.T. indicates the corresponding number in the User Table.

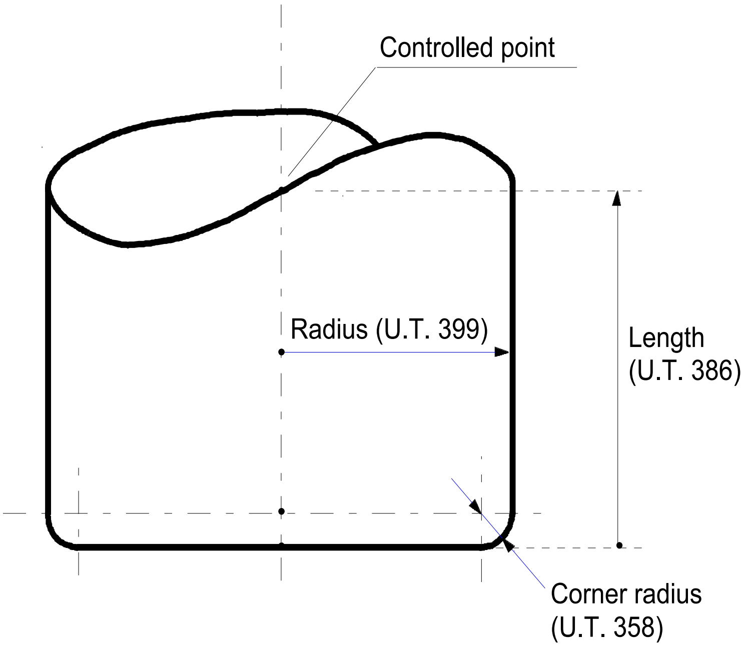


Fig. 2.7 Toroidal tool

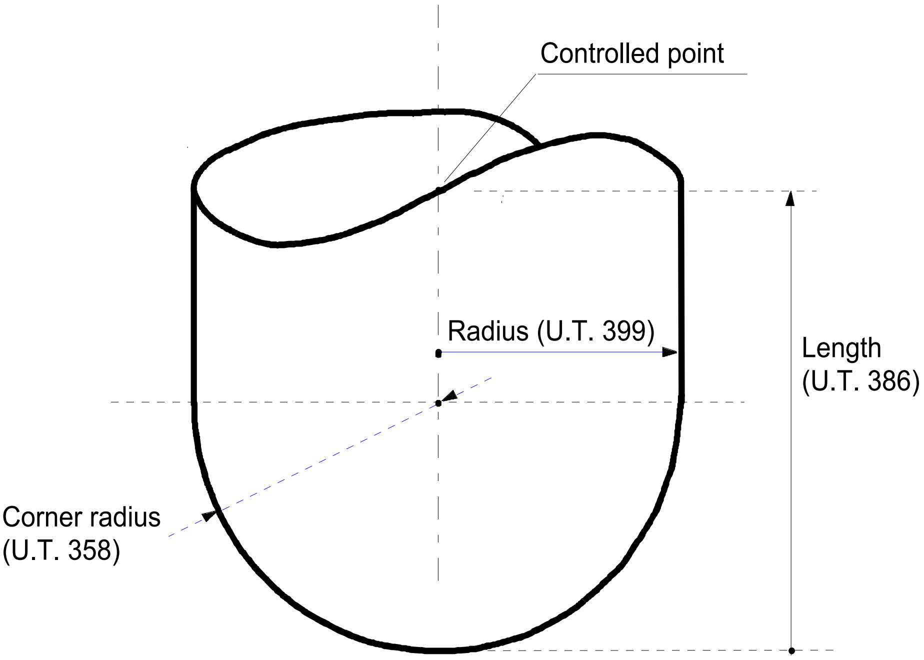


Fig. 2.8 Spherical tool

TCP Table for generic 5-axis machines

Variable	User Table	Record in PLUS User Table	Part Program	Meaning	TCP
Offset mode code	385	Record 97 - Var 1	L384	Defines the offset calculation mode according to the machine configuration. **NOTE:** Must be set with value = 6.	1 3 5
Tool length	386	Record 97 - Var 2	L385	Defines the length of the current mounted tool.	1 5
Active tool holder	387	Record 97 - Var 3	L386	Not used.	
Parameter A	388	Record 97 - Var 4	L387	Not used.	
Horizontal head offset	389	Record 98 - Var 1	L388	Not used.	
Horizontal head rotation	390	Record 98 - Var 2	L389	Not used.	
Vertical head rotation (figure 2.2)	391	Record 98 - Var 3	L390	Not used.	
1st linear axis ID	392	Record 98 - Var 4	L391	ID of the axis to be used as abscissa in the offset algorithm.	1 3 5
2nd linear axis ID	393	Record 99 - Var 1	L392	ID of the axis to be used as ordinate in the offset algorithm.	1 3 5
3rd linear axis ID	394	Record 99 - Var 2	L393	ID of the axis to be used as vertical axis in the offset algorithm.	1 3 5
Horizontal head ID	395	Record 99 - Var 3	L394	ID of the axis on which the horizontal head is mounted. If this ID=0, the axis does not exist.	1 3 5
Vertical head ID	396	Record 99 - Var 4	L395	ID of the axis on which the vertical head is mounted. If this ID=0, the axis does not exist.	1 3 5
Horizontal head inclination	397	Record 100 Var 1	L396	Not used.	
Vertical head inclination	398	Record 100 Var 2	L397	Not used.	
Radius (figure 2.1)	399	Record 100 Var 3	L398	Defines the radius of the tool currently mounted on the machine, with reference to the active tool holder.	1 5
Angle (figure 2.1)	400	Record 100 Var 4	L399	Not used.	

Dynamics

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Dynamic mode	357	Record 90 - Var 1	L356	These fields may have the following values: 0 - Speeds and accelerations of linear axes are not limited, and may exceed the configured values *run time* 1 - Speeds of linear and rotary axes are limited *run time*, if necessary, in such as never to exceed the configured values. Accelerations of the linear and rotary axes are limited in advance so that overlapping of movements does not require accelerations in excess of those configured. 2 - Speeds and accelerations of the linear and rotary axes are limited in advance so that the speeds and accelerations calculated *run time* never exceed the configured values. In this case, if for example only the linear axis is moved, it remains limited unlike what happened in case number 1. It must be stressed that in case 2, the constant speed is guaranteed on the profile, which does not happen in case 1 if speed of an axis is cut *run time.*	1 3 5
Corner radius (figures 2.7 and 2.8)	358	Record 90 - Var 2	L357	Radius of the tool corner in cases of spherical or toroidal tools	1, 3, 5
Angle of contact a	359	Record 90 - Var 3	L358	Not used.	
Programming of "m", and "n"	360	Record 90 - Var 4	L359	Not used.	
Threshold angle	361	Record 91 - Var 1	L360	Not used.	
Rotary axis velocity	362	Record 91 - Var 2	L361	Not used.	
Interpolation mode	363	Record 91 - Var 3	L362	Specifies whether or not the rotary axes are interpolated in conjunction with the linear axes. If not interpolated, the rotary axes follow the linear axes moves. Allowed values are: 0 - Rotary axes interpolated in conjunction with linear axes 1 - Rotary axes follow linear axes Select mode 1 if the velocity on the profile should not be affected by rotary axes motion.	1 3 5
Interpolation type	364	Record 91 - Var 4	L363	Specifies whether the rotary axes must be interpolated only by the error or must be treated as linear axes. Allowed values are: 0 - Rotary axes interpolated as linear axes 1 - Rotary axes interpolated only by the error.	1 3 5

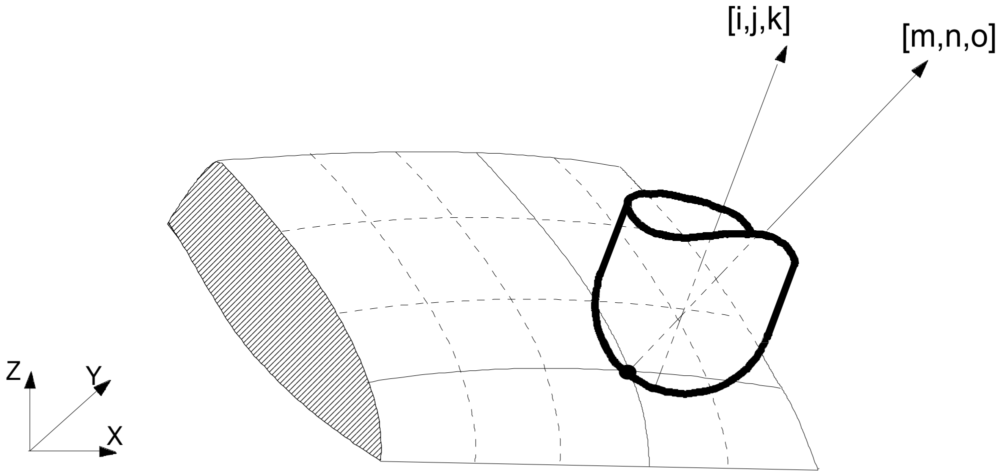
Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Integrator	365	Record 92 - Var 1	L364	With interpolation mode 1, this parameter defines whether acceleration/deceleration ramps will be used for rotary axes or whether velocities proportional to the linear axes moves will be used. Allowed values are as follows: 0 - No ramps 1 - Ramps If 1 is selected, the velocity diagram will be calculated using the configured parameters for rotary axes, irrespective of linear axes moves.	1 3 5
Offset mode	366	Record 92 - Var 2	L365	Defines the modalities of activation of the length corrector and the tool radius (and possibly the tool corner radius based on the TTR variable). 0 - values taken from the TCP table 1 - values with sign active in the system (length 1 and tool radius) 2 - absolute values active in the system (length 1 and tool radius) 3 - absolute value of length 1 only; the tool radius is considered = 0.	1 2 3 5
Minimum move	367	Record 92 - Var 3	L366	Not used.	
Travel limits control	368	Record 92 - Var 4	L367	Defines how the configured travel limits will be controlled. Allowed values are: 0 - travel limits controlled both in the program block (before execution) and at each point in real time. 1 - travel limits controlled both only at each point in real time. **NOTE:** Since part programs are typically referred to the tool tip, program block control is also referred to the tool tip. Real time control is referred to the center of rotation of the axes. When the system detects that the point is beyond the configured limit, it gives an error signal and locks the axes motion with deceleration.	1 3 5

Programming

With this type of TCP, for all the modes described below, both direction (orientation) of the tool and normal direction on the surface to be machined must be defined.

Two vectors are associated with each point programmed, one normal on the surface being machined (parameters m,n,o) and one representing the tool direction (parameters i,j,k).

The parameters m, n, o and i, j and k must be programmed in alphabetical order (e.g. mno, mo, no, ijk, ik, etc are correct; nmo, on, om, jik, ki, etc. are incorrect).



Syntax

[XYZ] [AB] [m n o] [i j k]

---

where:

XYZAB	movement of axes
mno	normal vector on surface being machined
ijk	vector representing tool direction

Characteristics:

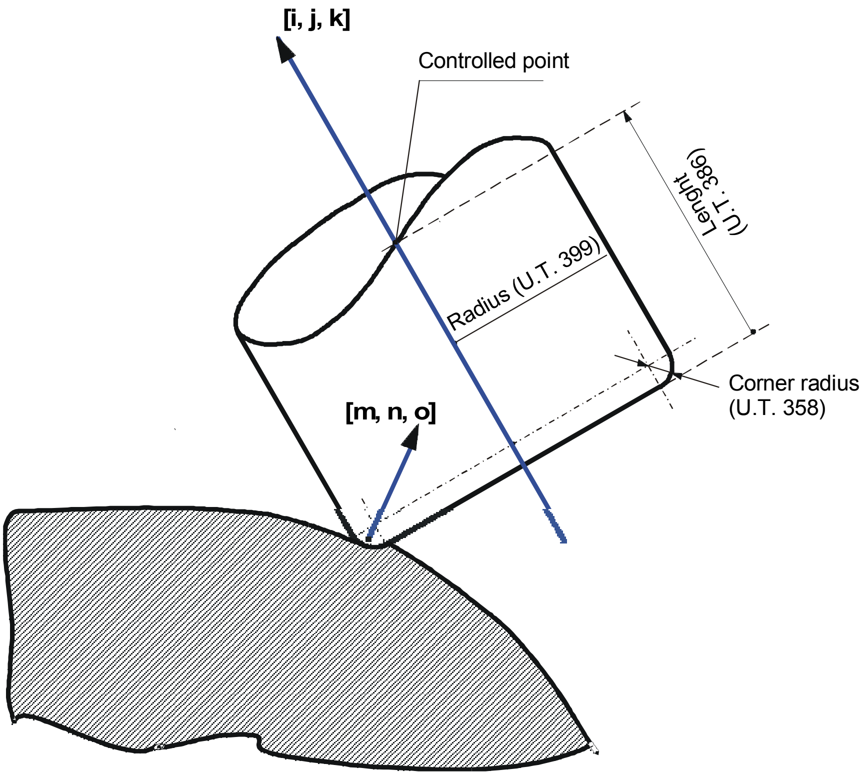
All the parameters are optional; if none are programmed, the previous programming values are reconfirmed.

Before activation of TCP, initial position of the two vectors must be defined, by programming values for them (this block does not cause any movement).
With TCP active, it is not possible to program *m n o* and *i j k* without a movement of the axes XYZAB; furthermore, the vector the define must not be null ([0,0,0]).

NOTE:
The RESET command resets the axes.

Example:

```
m..n..o..  i..j..k..            ;initial value of vectors
(TCP,....)                      ;TCP activation
X..Y..Z..  A..B.. m..n..o..  i..j..k.
X..Y..Z..  A..B.. m..n..o..
X..Z..  m..n..o..  i..j..k.
A..B.. m..n..o..
X..Y..  i..j..k.
.
.
.
.
(TCP)
```



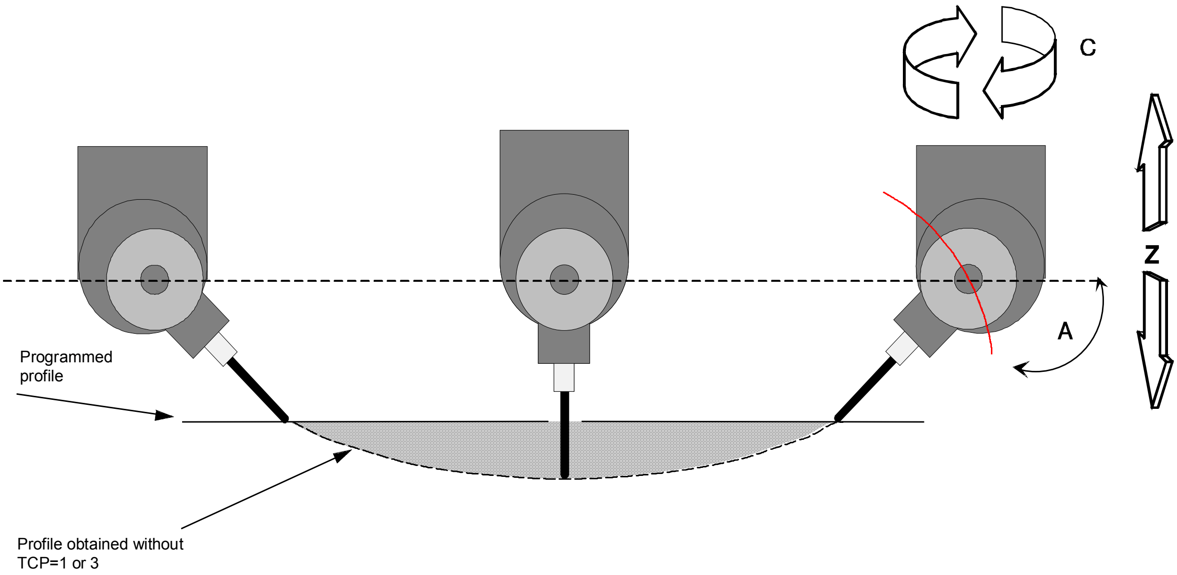
- **Mode 1 (TCP,1)**

With (TCP,1) programmed coordinates are referred to the part.
This mode must be used when the profile must be executed with both linear (1÷3) and rotary (1÷2) axes, the tool is expected to remain permanently in contact with the machining surface, and the linear axes coordinates are referred to the part.
To keep the tool tip on the profile defined by the linear axes, the system offsets displacements by automatically adjusting the position of the three linear axes to the rotary axes.

- **Mode 3 (TCP,3)**

This mode permits to use CAD programs in which the tool length, radius and corner radius are different from the ones with which the program (machine coordinates) was generated.
Tool length, radius and corner radius variances must be input via machine logic (refer to the $TCPWRT instruction in the PLUS Library Manual).

As with (TCP,1) in order to keep the tool tip on the profile defined by the linear axes, the system offsets displacements due to variations of length and radii by automatically adjusting the position of the three linear axes to the rotary axes.



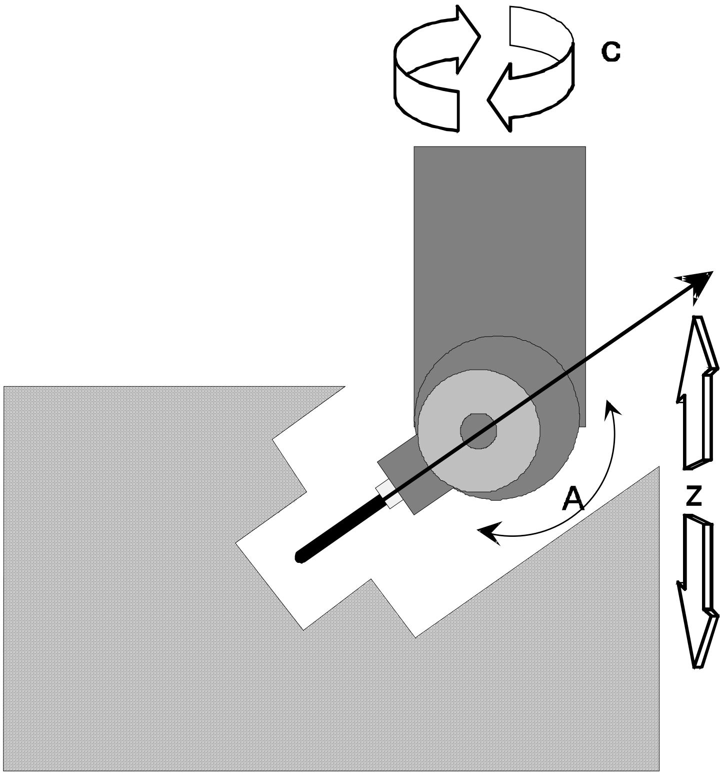
Profile obtained without TCP1 or 3 active

- **Mode 5 (TCP,5)**

This mode permits to execute moves in the tool direction.
When (TCP,5) is active the system generates a virtual axis whose name has been configured in AMP (VIRTUAL AXES section of the process configuration).
The virtual axis move generates a move of the linear axes in which the tool displaces in its direction, i.e. in the direction indicated by the vector [i,j,k].
(TCP,5) may be programmed both separately or in conjunction with the other axes, in which case (TCP,5) features add to (TCP,1) features. The move may also be made manually.

Mode 5 may be used for drilling inclined holes.

NOTE:
For more information about the MANUAL SETUP data entry or the $TCPWRT function please refer to the PLUS Library Manual and the User Manual.



Removing the tool with TCP,5

NOTE:
In all TCP modes, the machine logic may send to the interpolator in real time any variance of tool length, tool radius, or of the corner radius, (parameters 386, 399 and 358 in the User Table). This permits to offset tool wear during machining (refer to the description of the $TCPWRT function in the PLUS Library Manual).

TCP - Tool Center Point for machines with fixed tool and rotary table

This performance allow an easy programming on machines in which the tool is in a fixed position and the part moves by means of three axes, two linear ones and a rotary one solid to the linear ones. The system identifies in real time, according to the rotary axis position, the points that the axes must reach to satisfy the programmed profile. It is also possible to make the system calculate the rotary axis position and therefore to define the plane profile defined by the two linear axes. The activation of this algorithm is carried out directly in the program by means of the three-letters TCP.

The TCP mode is disabled with (**TCP**) or with RESET.

Syntax

(TCP[,n])

---

where:

n	Type of desired compensation (1, 2,)

Characteristics:

The TCP algorithm needs, for its functioning, of some information relative to the used tool, which have to be inserted into the “User Table” - abbr..U.T.

The “User Table” compilation may be executed by means of the proper editor, or by the machine logic (by means of proper logic functions), or also from part program by means of L variables.

It is possible to use a system configuration already pre-arranged contained in the files chargeable by means of "SETUP" of USER TABLE EDITOR.

NOTE:
For further information relative to the use of the “User Table” read the functions description from PLUS $TBLPUTD and $TBLGETD in the manual of PLUS library.

To use the same table from TABLE EDITOR environment see the User manual.

TCP Table for machines with fixed tool and rotary table

Variable	User Table	Record in PLUS User Table	Part Program	Meaning	TCP
Offset mode code	385	Record 97 - Var 1	L384	Defines the offset calculation mode according to the machine configuration. **NOTE:** Must be set with value = 8.	1 2
Tool length	386	Record 97 - Var 2	L385	Not used	
Active tool holder	387	Record 97 - Var 3	L386	Number of the active tool holder 1 ÷ 4	1 2
Parameter A	388	Record 97 - Var 4	L387	Not used.	
Horizontal head offset	389	Record 98 - Var 1	L388	Not used.	
Horizontal head rotation	390	Record 98 - Var 2	L389	Defines the rotation direction of the rotary axis (table) and has a value of: + 1 clock-wise - 1 anticlock-wise for positive programming	1 2
Vertical head rotation	391	Record 98 - Var 3	L390	Not used.	
1st linear axis ID	392	Record 98 - Var 4	L391	ID of the axis to be used as abscissa in the offset algorithm.	1 2
2nd linear axis ID	393	Record 99 - Var 1	L392	ID of the axis to be used as ordinate in the offset algorithm.	1 2
3rd linear axis ID	394	Record 99 - Var 2	L393	Not used.	1 2
Horizontal head ID	395	Record 99 - Var 3	L394	ID of the rotary axis (table)	1 2
Vertical head ID	396	Record 99 - Var 4	L395	Not used	
Horizontal head inclination	397	Record 100 Var 1	L396	Not used.	
Vertical head inclination	398	Record 100 Var 2	L397	Not used.	
Radius	399	Record 100 Var 3	L398	Not used.	
Angle	400	Record 100 Var 4	L399	It defines the angle between the tool center and the cutter contact point after having rotated the vertical head in order to have the tool orientated in y-direction	1 2

Tool holder 1

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B	369	Record 93 - Var 1	L368	Sign distance expressed in mm of the table centre from the tool tip as regards the abscissa axis.	1 2
Parameter C	370	Record 93 - Var 2	L369	Sign distance expressed in mm of the table centre from the tool tip as regards the ordinate axis.	1 2
Parameter D	371	Record 93 - Var 3	L370	Not used	
Offset vertical head	372	Record 93 - Var 4	L371	Not used	

Tool holder 2

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B (figures 2.1, 2.3)	373	Record 94 - Var 1	L372	As in tool holder 1	1 2
Parameter C (figures 2.1, 2.2)	374	Record 94 - Var 2	L373	As in tool holder 1	1 2
Parameter D	375	Record 94 - Var 3	L374	Not used	
Vertical head offset	376	Record 94 - Var 4	L375	Not used	

Tool holder 3

Variable	User Table	Record in PLUS user table	Part Program.	Meaning	TCP
Parameter B	377	Record 95 - Var 1	L376	As in tool holder 1	1 2
Parameter C	378	Record 95 - Var 2	L377	As in tool holder 1	1 2
Parameter D	379	Record 95 - Var 3	L378	Not used	
Offset vertical head	380	Record 95 - Var 4	L379	Not used	

Tool holder 4

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Parameter B	381	Record 96 - Var 1	L380	As in tool holder 1	1 2
Parameter C	382	Record 96 - Var 2	L381	As in tool holder 1	1 2
Parameter D	383	Record 96 - Var 3	L382	Not used	
Offset vertical head	384	Record 96 - Var 4	L383	Not used	

Dynamics

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Dynamic mode	357	Record 90 - Var 1	L356	These fields may have the following values: 0 - Speeds and accelerations of linear axes are not limited, and may exceed the configured values *run time* 1 - Speeds of linear and rotary axes are limited *run time*, if necessary, in such as never to exceed the configured values. Accelerations of the linear and rotary axes are limited in advance so that overlapping of movements does not require accelerations in excess of those configured. 2 - Speeds and accelerations of the linear and rotary axes are limited in advance so that the speeds and accelerations calculated *run time* never exceed the configured values. In this case, if for example only the linear axis is moved, it remains limited unlike what happened in case 1. It must be stressed that in case 2, the constant speed is guaranteed on the profile, which does not happen in case 1 if speed of an axis is cut *run time.*	1 2
Corner radius[1]	358	Record 90 - Var 2	L357	Not used	
Angle of contact a	359	Record 90 - Var 3	L358	Not used	
Programming of "m", "n" and "o"	360	Record 90 - Var 4	L359	Not used	
Threshold angle (figure 2.4)	361	Record 91 - Var 1	L360	Angle beyond which the system adds a horizontal rotary axis move --after the element has been completed-- in order to position the tool perpendicular to the subsequent element. Below this angle, the horizontal rotary axis move starts when the element is completed and continues during execution of the subsequent element.	2
Rotary axis velocity	362	Record 91 - Var 2	L361	Velocity expressed in degrees per minute at which the horizontal rotary axis moves when it is automatically positioned between two part program blocks. If it is 0 the system uses the programmed feedrate.	2

Dynamics (continued)

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Interpolation mode	363	Record 91 - Var 3	L362	Specifies whether or not the rotary axes are interpolated in conjunction with the linear axes. If not interpolated, the rotary axes follow the linear axes moves. Allowed values are: 0 - Rotary axes interpolated in conjunction with linear axes 1 - Rotary axes follow linear axes Select mode 1 if the velocity on the profile should not be affected by rotary axes motion.	1 2
Interpolation type	364	Record 91 - Var 4	L363	Specifies whether the rotary axes must be interpolated only by the error or must be treated as linear axes. Allowed values are: 0 - Rotary axes interpolated as linear axes 1 - Rotary axes interpolated only by the error.	1 2
Integrator	365	Record 92 - Var 1	L364	With interpolation mode 1, this parameter defines whether acceleration/deceleration ramps will be used for rotary axes or whether velocities proportional to the linear axes moves will be used. Allowed values are as follows: 0 - No ramps 1 - Ramps If 1 is selected, the velocity diagram will be calculated using the configured parameters for rotary axes, irrespective of linear axes moves.	1 2
Offset mode	366	Record 92 - Var 2	L365	Not used	
Minimum move	367	Record 92 - Var 3	L366	Specifies the minimum angle beyond which the system automatically generates a move of the horizontal rotary axis in order to position the tool perpendicular to the subsequent element.	2

Dynamics (continued)

Variable	User Table	Record in PLUS user table	Part Program	Meaning	TCP
Travel limits control	368	Record 92 - Var 4	L367	Defines how the configured travel limits will be controlled. Allowed values are: 0 - travel limits controlled both in the program block (before execution) and at each point in real time. 1 - travel limits controlled both only at each point in real time. **NOTE:** Since part programs are typically referred to the tool tip, program block control is also referred to the tool tip. Real time control is referred to the center of rotation of the axes. When the system detects that the point is beyond the configured limit, it gives an error signal and locks the axes motion with deceleration.	1 2

NOTE:
In the above tables distances are expressed in mm. They must be intended in the configured length unit.

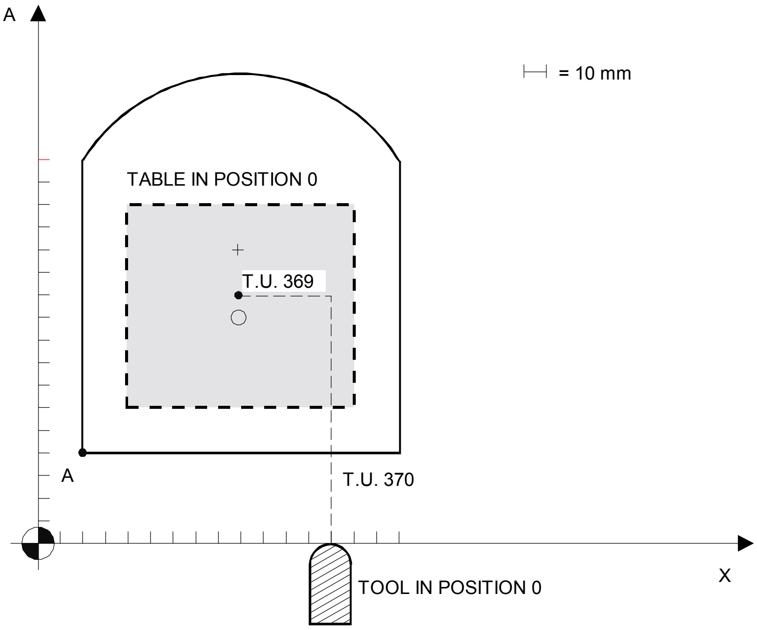
Programming

The rotary table virtualization allows the programming of machines with tool in fixed position and the part moving by means of three axes, two linear ones and one rotary solid to the two linear ones. It is activated by commands (TCP, 1) or (TCP, 2).
By programming (TCP, 2) the automatic insertion of rotary axis is obtained. (TCP, 1) and (TCP, 2) performances and characteristics described for 5-axis machines are valid also for this kind of compensation.
For a correct programming on rotary table machines, the following must be taken into consideration:

a) the system considers to have, at the end of the homing cycle, coincident tool tip and table centre. In case this situation does not happen, the values representing the distance of the tool centre from the tool tip must be defined in the TCP table (T.U.369 e T.U. 370)

b) the profile must be defined as regards an origin coincident with the table centre, therefore, as described in the previous point, if table centre and tool tip do not coincide, the origin must be defined with the two values of abscissa and ordinate equal to the phase displacement defined in the TCT table (T.U. 369 e T.U. 370).

Example:



T.U. 369 = - 40	Distance, as regards X, of the table centre from the tool tip
T.U. 370 = + 110	Distance, as regards Y, of the table centre from the tool tip

```
ORIGIN 1 (POINT O)  X-40        Y+110
(UOT,1,X-70,Y-70)               ; POINT A
GX-10Y-10
(TCP,2)
G1 Y0  F2000
X+140
Y+130
G3 X0  Y130  I70  J90
G1Y-10
(TCP)
```

TCP - Tool Center Point for machines with Prismatic Head

This function is used to program a machine with 5 axes (3 linear and 2 rotary axes) with reference to the tip of the tool rather than the rotation centre of the axes (centre of the head). It differs from the standard Double Twist Head, in that the vertical rotary axis rotates in a plane not orthogonal to the plane of the horizontal rotary axis. The activation of this algorithm is performed directly in the program through the TCP triliteral. The TCP mode is disabled with (TCP) or RESET.

It is possible to program this type of head by regarding it as a standard Double Twist Head (rotation of an axis in the horizontal plane and the other axis in a vertical plane), in this case, the system sees to the adjustments needed.

The TCP mode is disabled with **(TCP)** or RESET.

Syntax

(TCP[,n])

---

where:

n Type of compensation desired (1÷5); these modes are described in the previous pages.

Characteristics:

To work, the algorithm of TCP requires some data on the tool being used. The head managed accepts a single vertical or angular tool. The configuration parameters must be evaluated after positioning the head so that the tool is arranged vertically along the Z- direction and the rotation axis of B is parallel to the XZ plane as shown in figures 2.9, 2.10, 2.11.
These parameters must be entered in the "User Table". The User Table can be compiled by means of the ad hoc editor or through the machine logic (through the ad hoc logic functions), or by means of the part program through the L variables. It is possible to use a configuration already predisposed by the system, which is contained in files that can be loaded through the "SETUP" of the USER TABLE EDITOR. These files are called TCP7_ITA and TCP7_ENG.

NOTES:
For further information on the use of the User Table from PLUS, read the description of the $TBLPUTD and $TBLGETD functions in the manual of the PLUS library.

For the use of the same table in the TABLE EDITOR environment, see the User Manual.

TCP table for machines with "Prismatic" head

Variable	User table	PLUS user table records	Part Program	Meaning of the variable	TCP
Type of compensation	385	Record 97 - Var 1	L384	Defines the compensation calculation mode as a function of machine configuration. **NOTE:** set a value of 7	1 2 3 5
Tool length ①	386	Record 97 - Var 2	L385	Defines the length of the tool currently fitted to the machine.	1 2 5
Free	387	Record 97 - Var 3	L386		
Horizontal head offset ①	388	Record 97 - Var 4	L387	It is the offset value, expressed in degrees, to be associated with the axis of the horizontal head in order to have the plane containing the tool coincide with the XZ plane of the machine. To determine this offset rotate the horizontal axis so that the plane containing the tool coincides with the XZ plane and that, after this rotation, a positive movement of the tilting head aims the tool towards Z-. At this point, take the value displayed on the screen, multiply it by the direction of rotation of the horizontal head and change the sign.	1 2 3 4 5
Tilting head offset ①	389	Record 98 - Var 1	L388	It is the offset value, expressed in degrees, to be associated with the axis of the tilting head in order to position the tool along the Z- direction. To determine this offset rotate the tilting axis to move it in the position previously specified. At this point, take the value displayed on the screen, multiply it by the direction of rotation of the tilting head and change the sign.	1 2 3 4 5
Direction of horizontal head ① (figure 2.11)	390	Record 98 - Var 2	L389	Defines the direction of rotation of the horizontal head viewed from the top, and it is: +1 for the clockwise direction - 1 for the counter clockwise direction in positive programming.	1 2 3 4 5
Direction of tilting head ① (figure 2.10)	391	Record 98 - Var 3	L390	Defines the direction of rotation of the tilting head viewed from the left side and it is: +1 for the clockwise direction - 1 for the counter clockwise direction in positive programming.	1 2 3 4 5

From this point on, it continues as standard” Double Twist” table.

Head Table

Variable	User table	PLUS user table records	Part Program	Meaning	TCP
Parameter A ①	369	Record 93 - Var 1	L368	Distance along the first axis (X) between the control point **Pc** and the rotation point **Pr**. This distance must be expressed in mm and has a positive value when the transition from Pc to Pr takes place along the negative direction of the axis.	1 2 5
Parameter B ① (figures 2.10 and 2.11)	370	Record 93 - Var 2	L369	Distance along the second axis (Y) between the control point **Pc** and the rotation point **Pr**. This distance must be expressed in mm and has a positive value when the transition from Pc to Pr takes place along the negative direction of the axis.	1 2 5
Parameter C ① (figures 2.9 and 2.11)	371	Record 93 - Var 3	L370	Distance along the third axis (Z) between the control point **Pc** and the rotation point **Pr**. This distance must be expressed in mm and has a positive value when the transition from Pc to Pr takes place along the negative direction of the axis.	1 2 5
Parameter D ① (figures 2.9 and 2.11)	372	Record 93 - Var 4	L371	Distance along the first axis (X) between the rotation point **Pr** and the tool attachment point **Pu**. This distance must be expressed in mm and has a positive value when the transition from Pr to Pu takes place along the negative direction of the axis.	1 2 5
Parameter E ①	373	Record 94 - Var 1	L372	Distance along the second axis (Y) between the rotation point **Pr** and the tool attachment point **Pu**. This distance must be expressed in mm and has a positive value when the transition from Pr to Pu takes place along the negative direction of the axis.	1 2 5
Parameter F ① (figures 2.9 and 2.11)	374	Record 94 - Var 2	L373	Distance along the third axis (Z) between the rotation point **Pr** and the tool attachment point **Pu**. This distance must be expressed in mm and has a positive value when the transition from Pr to Pu takes place along the negative direction of the axis.	1 2 5
Inclination (figure 2.9)	375	Record 94 - Var 3	L374	Inclination of the axis around which the tilting axis rotates relative to the machine vertical.	1 2 3 5

Tilting axis management mode	376	Record 94 - Var 4	L375	Defines the programming mode of the tilting axis and is: With **0** it is programmed in terms of transducer dimensions (degrees relative to its axis of rotation). With **1** it is programmed in terms of transducer dimensions (as in 0), but the horizontal rotary axis is automatically oriented by the system so that, no matter how the tilting axis is moved, the tool remains in the plane defined by the dimensions programmed on the horizontal rotary axis. With **2** the inclination of the tilting axis is not programmed with reference to the transducer but rather in terms of inclination in space; 0 is defined with the tool aimed towards Z-; at all events, the horizontal rotary axis is oriented automatically. In the latter case, the programming of the head is the same as the programming of a standard double twist head.	1 2 3 5
+ operational limit of tilting axis	377	Record 95 - Var 1	L376	It is used solely for programming mode 2. It is the positive limit programmable. Its maximum value is determined on the basis of the value given in the "inclination" field as: acos(2*sin2(90-incl)-1). It is important to set the lower limit of this value, as around it the speed of the tilting axis approaches ∞ and thereby generates a servo error.	1 2 3 5
- operational limit of tilting axis	378	Record 95 - Var 2	L377	It is used solely for programming mode 2. It is the negative limit programmable. Its minimum value is determined on the basis of the value given in the "inclination" field as: -acos(2*sin2(90-incl)-1). It is important to set the upper limit of this value, as around it the speed of the tilting axis approaches ∞ and thereby generates a servo error.	1 2 3 5
Offset of tilting axis ①	379	Record 95 - Var 3	L378	It is used solely for programming mode 2. It is the offset value, expressed in degrees, to be associated with the tilting axis in order to position the tool along the Z- direction. To determine this offset rotate the tilting axis so that the tool is moved to the position specified earlier. At this point, take the value displayed, multiply it by the direction of the tilting axis and change the sign.	1 2 3 5

Direction of tilting axis ①	380	Record 95 - Var 4	L390	It is used solely for programming mode 2. It defines the direction of rotation of the tilting axis viewed from the left side and is: +1 for the clockwise direction - 1 for the counter clockwise direction in positive programming	1 2 3 4 5

ANGULAR TRANSMISSION

A PRISMATIC head with angular transmission (figure 2.12) must be characterised by considering initially the tool in a vertical position and by determining the parameters of the head movements accordingly. After that, with the B rotation axis parallel to the XZ plane and starting from the position of the tool along the Z- direction, arrange the angular transmission in the appropriate working position and determine the other four tool parameters.

An angular head, in fact, is always identified by four parameters, i.e., two lengths and two angular measurements. The two lengths (offset in the direction of the Z axis) reflect the following distances: from the centre of the head to the centre of the angular transmission (L2) and from the centre of the transmission to the centre of the relative spindle (L1). The two angular measurements correspond to the upward and transverse movements of the tool holder: offset around X (beta) and offset around Z (alpha)

Angular transmission table

Variable	User table	PLUS user table records	Part Program	Meaning	TCP
Length 1 ① (figure 2.12)	381	Record 96 - Var 1	L380	Defines length L1 in mm	1 2 5
Length 2 ① (figure 2.12)	382	Record 96 - Var 2	L381	Defines length L2 in mm	1 2 5
Angle alpha ① (figure 2.12)	383	Record 96 - Var 3	L382	Defines the phase displacement angle α, in degrees, of the transmission around the Z axis. (-180° σ α σ 180°).	1 2 5
Angle beta ① (figure 2.12)	384	Record 96 - Var 4	L383	Defines the inclination β, in degrees, of the transmission relative to the Z-axis. (-90° σ β σ 90°).	1 2 5

① This parameter can be varied continuously through lowercase (tcp,n) programming.

For the compilation of the Dynamic table see the table for the standard “Double Twist” machine.

The figures below illustrate the compensation characterisation parameters applied to a machine shown in schematic front, side and top views.

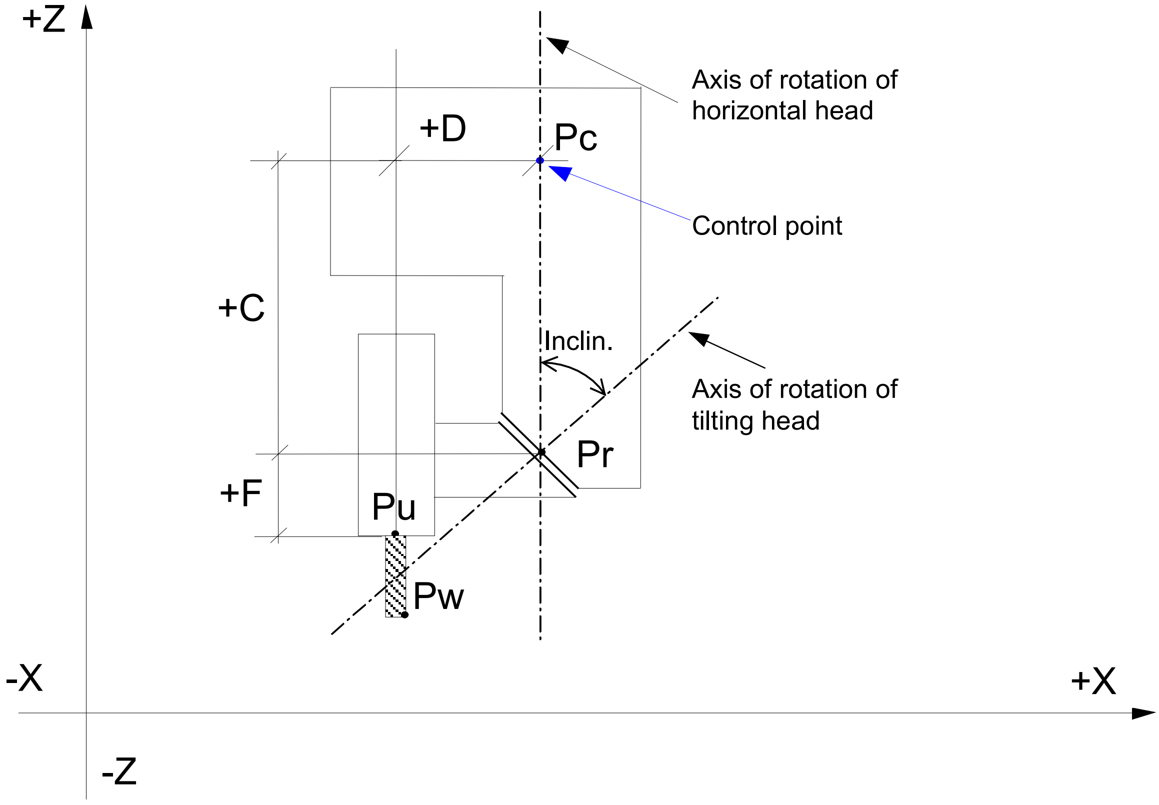

+Z
+D
Pc
Axis of rotation of horizontal head
Control point
+C
Inclin.
Axis of rotation of tilting head
Pr
+F
Pu
Pw
-X
+X
-Z

Fig. 2.9 Front view

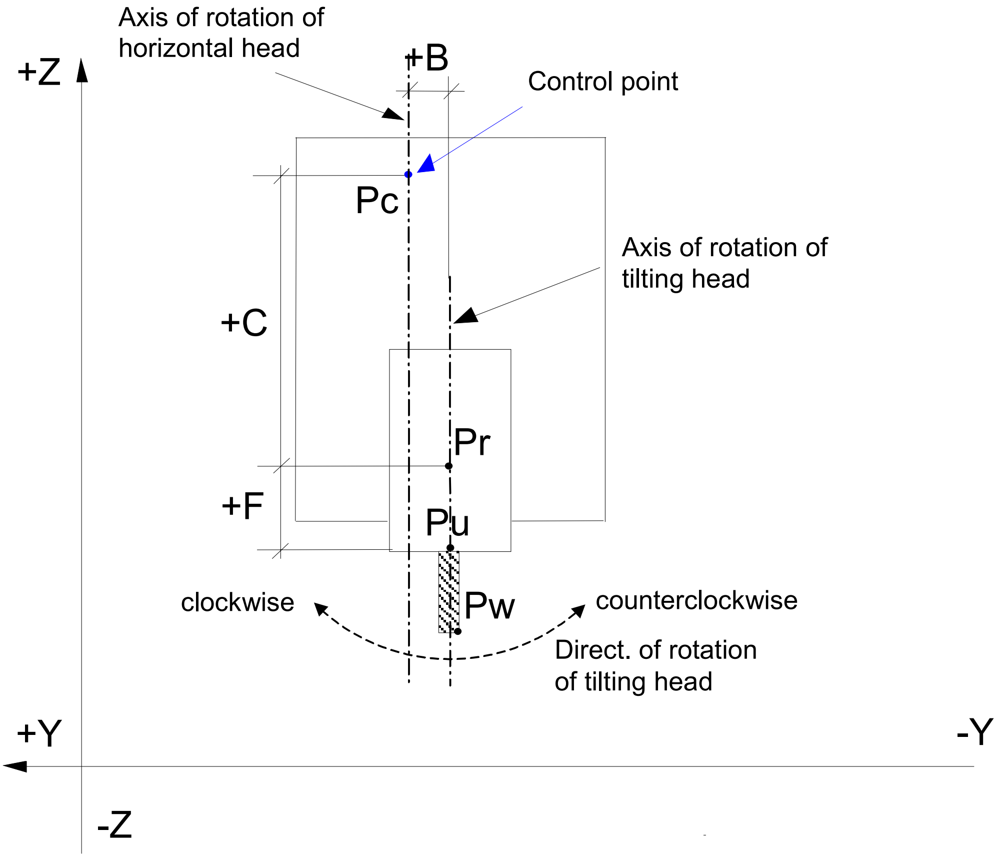

Axis of rotation of horizontal head
+Z
+B
Control point
Pc
Axis of rotation of tilting head
+C
Pr
+F
Pu
clockwise
Pw
counterclockwise
Direct. of rotation of tilting head
+Y
-Y
-Z

Fig. 2.10 Side view

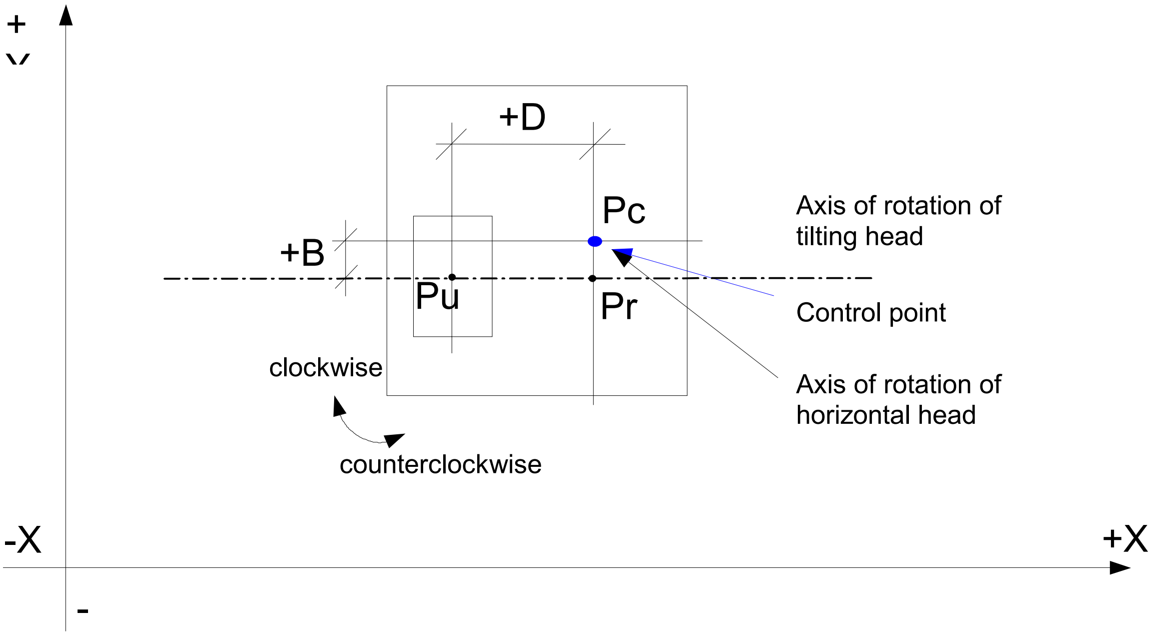

+
+D
Pc
Axis of rotation of tilting head
+B
Pu
Pr
Control point
clockwise
Axis of rotation of horizontal head
counterclockwise
-X
+X
-

Fig. 2.11 Top view

An angular tool, as shown in the figure, is characterised by four parameters: length **L1**, length **L2**, phase displacement angle α around axis Z, inclination β of the tool relative to axis Z-.

ANGULAR TOOL

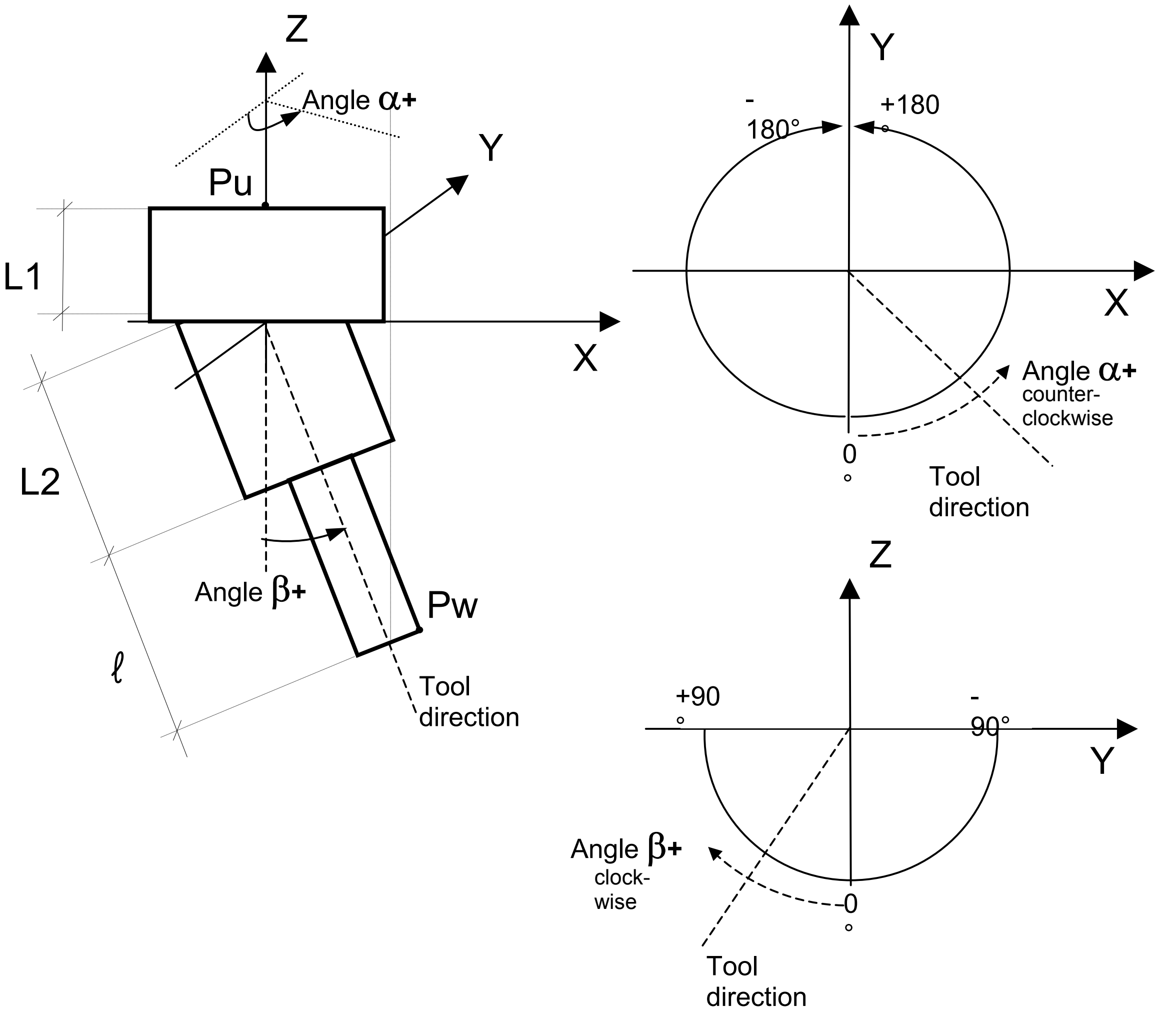


Fig. 2.12

Programming

With the same functional specifications it is possible to use all the TCPs admissible for the double twist head. It is also possible to program both using the m,n,o versors normal to the piece and the contact angles, m and n.

TCP on multi-processor

The TCP feature is now extended to 4 processes, each process will use a setup area of its own in the User tables as shown in the table below.

Process	Tabble	Plus Table Record	Part Program LVariable
1	TCP Table	Record 97	L384 ÷ L399
	Tool-holder 1	Record 93	L368 ÷ L371
	Tool-holder 2	Record 94	L372 ÷ L375
	Tool-holder 3	Record 95	L376 ÷ L379
	Tool-holder 4	Record 96	L380 ÷ L383
	Dynamics	Record 90	L356 ÷ L367
2	TCP Table	Record 86	L340 ÷ L355
	Tool-holder 1	Record 82	L324 ÷ L327
	Tool-holder 2	Record 83	L328 ÷ L331
	Tool-holder 3	Record 84	L332 ÷ L335
	Tool-holder 4	Record 85	L336 ÷ L339
	Dynamics	Record 79	L312 ÷ L323
3	TCP Table	Record 75	L296 ÷ L311
	Tool-holder 1	Record 71	L280 ÷ L283
	Tool-holder 2	Record 72	L284 ÷ L287
	Tool-holder 3	Record 73	L288 ÷ L291
	Tool-holder 4	Record 74	L292 ÷ L295
	Dynamics	Record 68	L268 ÷ L279
4	TCP Table	Record 64	L252 ÷ L267
	Tool-holder 1	Record 60	L236 ÷ L239
	Tool-holder 2	Record 61	L240 ÷ L243
	Tool-holder 3	Record 62	L244 ÷ L247
	Tool-holder 4	Record 63	L248 ÷ L251
	Dynamics	Record 57	L224 ÷ L235

END OF CHAPTER

Chapter 3

PROGRAMMING TOOLS AND TOOL OFFSETS

This chapter explains how to program tools and tool offsets. All the functions described in this chapter must be handled by the machine logic.

WARNING

The system integrator develops the interface between the control and the machine tool, as well as the application-specific T and M functions and also of the code requalifying functioning and the presetting: RQT and RQP.

For more information about these T functions you must refer to the machine tool documentation provided by the machine manufacturer.

You can also find further information about M functions in Chapter 6.

T address for programming tools

The T address defines the tool and the tool offset used in a given machining process.

Syntax

T [*tool*], [.] [*tool offset*]

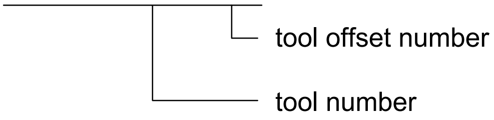
where:

tool	is the tool number. This can be programmed explicitly (with an integer) or implicitly with a local or system variable.
tool offset	is the tool offset number. It can be an integer or an E parameter.

Characteristics:

Allowed values for both parameters ranges from 0.0 to 999999999999.300. The meaning of the 15 digits is as follows:

999999999999.300



Examples:
The following examples show different ways of programming tools and offsets.

T1	Selects tool 1 and the tool offset defined in the tool table.
T1.1	Selects tool 1 and tool offset 1.
T1.0	Selects tool 1 without tool offset.
T.0	Removes the tool offset from the active tool.
T0	Disables the active tool and tool offset.
T.1	Enables tool offset 1 with the active tool.

T address for multi-tool programming

The T address defines the tools to be used simultaneously for a given machining process.

Sintax

T [*master*] [.] [*tool offset*] [*/*{*slave* I] [first_*slave, last slave*]I **{***variable, num_variables***}**}]

where:

master	Is the number of the tool. This can be an integer or a local or system variable between 0 and 999999999999.
tool offset	Is the number of the tool offset associated with the master tool. This can be an integer or a local or system variable between 0 and 300.
slave	Is the number of the tool. This can be an integer or a local or system variable.
first_slave	Is a tool number representing the first of a series of tools. This can be an integer or a local or system variable.
last_slave	Is a tool number representing the last of a series of tools. This can be an integer or a local or system variable.
variable	Is a local or system variable containing the first of a series of tools.
num_variable	Is an integer or a local or system variable representing the number of variables to take into consideration starting from "variable".

Characteristics:

Multi-tool programming is used on perforating machines.

The management of tools associated with the T code is handled by the logic of the machines to which the values of the programmed tools are sent..

The programmable values for the slave tool codes vary from 0 to 65535. The maximum number of slave tools which can be simultaneously programmed is 60.

As you can see from the T code sintax, the list of tools to be used can be specified in three different formats:

1. Single Format

Examples:

T1.2/ 50	tools	1, 50
	offset	2
T1.2 /20,33,45,46	tools	1,20,33,45,46
	offset	2

2. Numerical Sequence Format

This simplifies programming of multiple tools by using sequential code.

Examples:

T1.3 /[30, 35]	equivalent to T1.3 /30,31,32,33,34,35
T1.3 /[56, 51]	equivalent to T1.3 /56,55,54,53,52,51
T1.3 /[50, 52], [10,13]	equivalent to T1.3 /50,51,52,10,11,12,13

As you can see, the starting tool number can be > or < the final tool number. In the first instance the tools codes are considered in ascending order and in the second instance they are considered in descending order.

3. Variable Sequence Format

This is based on an array of variables from which to select the tool numbers.

Example:

E0 = 1, 30, 45	load in E0, E1, E2 the values 1, 30, 45
T1.2 /{ E0, 3 }	equivalent to T1.2 /E0, E1, E2 & to T1.2 /1, 30, 45

E0 = 1, 30, 45	load in E0, E1, E2, E3 the values 1, 30, 45
SN0 = 4, 77	load in SN0, SN1 the values 4, 77
SN4 = 3	load in SN4 the value 3
T1.2 /{ E0, SN4 }, { SN0,2 }	equivalent to T1.2 /E0, E1, E2, SN0, SN1 & to T1.2 /1,30,45,4,77

The three formats shown above may be mixed.

Example:

E0 = 29, 56	load in E0, E1 the values 29, 56
SN6 = 2	load in SN6 the value 2
T1.3 /[7, 10], 15, { E0, SN6 }	equivalent to T1.3 /7,8,9,10,15,29,56

h address

The h address permits to apply a tool offset both in point-to-point (G29) and in continuous (G27-G28) modes.

Syntax

---

h [*tool offset*]

---

where:

tool offset	is the tool offset number. It can be an integer or a local or system variable between zero and 300.

Characteristics:

The "h" address must be programmed by itself in one block.

An "h" address disables the active offsets programmed with a "T" command.
The axes to which *tool offset* is applied are those programmed in a "T *tool.offset"* command.

The *offset* values are applied to the axes when the system reads the "h" address in the part program. The "h" address must not be synchronised either with the logic or with the axes moves.

IMPORTANT

If “h” is not synchronized, it is displayed when it is read by the part program and not when it is implemented.
The same occurs in the case of synchronous uses of “h”, as in G96. In this case, the application of “h” must be synchronized with “#” for the spindle revolutions to function correctly.

If you program "h" without *tool offset* or with *tool offset=0,* the value programmed with the previous

"h *tool offset"* command. This value may be re-established with a RESET command.

IMPORTANT

The machine logic decides whether or not to apply offset values after a RESET.

IMPORTANT

The use of "h" and **T** for offsets in the same part program can cause problems if it is not handled correctly by the machine logic. It is recommended that only one of these modes be used at the same time.

The offset values programmed with h address are usually displayed in the field reserved for the T address.

IMPORTANT Offsets may be introduced with an h address only if the default configuration of the offsets table is used.

The offset table fields from which offset values are read are as follows:

Offset value	Offset table fields
Length 1	Length 1 + Current requalification length 1
Length 2	Length 2 + Current requalification length 2
Diameter	Diameter + Current requalification diameter

Example 1:

```
.....
.....
.....
T[n].x          programs a tool and a tool offset
.....
.....
hy              cancels the x tool offset and enables the y tool offset
.....
.....
```

Example 2:

```
.....
.....
.....
hy              enables y tool offset
.....
.....
T[n].x          cancels the y tool offset enabled by h
.....
.....
```

Example 3:

```
.....
.....
.....
hx
.....
.....
hy              cancels the programmed x tool offset and enables y
.....
```

TTR – Toroidal Tool Radius

This process variable makes it possible to enable the management of the tool corner radius when using spherical or toroidal tools.

Syntax

TTR = value

---

where:

value	This value can be:	
	0	Disables the management of the tool corner radius
	1	Enables the management

NOTE

The moment the tool corner radius management is enabled, the length 2 field (+ requalification length and max. length) is used by the system to define the tool tip radius. In this connection, the length is no longer applied as a length corrector.

The tool corner radius is used only within the TCP and the HSM for programming a T or an h.

AXO - Axis Offset Definition

This command makes it possible to associate the length offset values contained in the offset table to the machine axes.

Syntax

(AXO,[[-] *axis name1*][**,** [-] *axis name2*]**)**

---

where:

axis name1	name of the axis associated to length offset 1 in the offsets table. If the "-" sign is written before the name, the offset value is used with the inverse sign. If omitted, the length 1 of the offset is not associated with any axis.
axis name2	name of the axis associated to length offset 2 in the offsets table. If the "-" sign is written before the name, the offset value is used with the inverse sign. If omitted, the length 2 of the offset is not associated with any axis.

Characteristics:

The default association between length offsets and characterised axes is made in the AXIS GENERAL INFO page of the AMP utility.

This association can be changed with the AXO command.
The preset values introduced in the offset tables always have a positive sign. The AXO command enables these values to be associated with axes bearing negative signs.

The following are two examples of length offset values applied with and without AXO commands:

Example 1:

.
.
.

N1 T1.4 M6	activates length 1 of offset 4 on the axis associated to length 1 in the offset characterisation phase.

.
.

N100 T0 M6	disables the length offset value applied to the axis.

Example 2:

.	
.	
.	
N1 (AXO,-X,Z)	associates X to length offset 1 with negative sign and Z to length offset 2 with positive sign
.	
.	
N50 T1.4 M6	applies length offset values 1 and 2 from offset 4 to axes X and Z as defined in the AXO command. Length offset 1 will be applied to the X axis with negative sign.
.	
.	
N100 T0 M6	disables the length offset applied to axes X and Z.

Example 3:

.	
.	
N1 (AXO,,Z)	associates the length 2 of the offset to axis Z.

IMPORTANT

The system RESET command or the selection of a new part program re-establish the axis/offset default association characterised with AMP.

RQT (RQU) - Requalifying Tool Offset

The RQT command requalifies the length and diameter dimensions memorised in the offset tables.

Syntax

(RQT,*tool***,***offset* [**,L**..][**,I**..][**,d**..]**)**

where:

tool	Is the tool number.
offset	Is the number of the offset to be requalified. The offset number is a value between 1 and 300.
L	Is the value to be added to offset length 1.
I	Is the value to be added to offset length 2.
d	Is the offset increment to be added to the diameter.

Characteristics:

The values can be either numbers or contents in the variables.

You must specify the length and diameter increments in the RQT command with the measuring unit configured in the system (inches or millimetres, G70/G71). These values cannot be associated to a scale factor (SCF).

Examples:

(RQT,10,1,L E40,d E41)	This block requalifies tool 10, offset 1. The length 1 increment is contained in E40 and the diameter increment is contained in E41.
(RQT,10,1,L E50,I E51)	This block requalifies tool 10, offset 1. The length 1 increment is contained in E50 and the length 2 increment is contained in E51.

RQP - Requalifying Tool Offset

The RQP command requalifies and presets a specific tool offset according to programmed length and diameter dimensions. When the control executes this command, it writes the corresponding dimensions in the tool offset table by adding the specified length and diameter values. All the offset values are reset.

Syntax

(RQP,*tool***,***offset* [,**L**..] [,**l**..][,**d**..]**)**

where:

tool	Is the tool number.
offset	Is the number of the offset to preset. It is a value between 1 and 300. The upper limit of the offset number depends on the number of records in the tool offset file.
L	Is the length increment to write to length offset 1.
l	Is the length increment to write to length offset 2.
d	Is the diameter increment to write to the offset field.

Characteristics:

The values can either be some numbers or contents in the variables.

You must specify the values of length and diameter increments in the RQP command with the measuring unit configured in he system (inches or millimetres, with G70/G71). These values cannot be associated to a scale factor (SCF).

Examples:

(RQP,10,1,L E40 ,d E41)	This block presets offset 1 of tool 10. The value of length 1 is contained in E40 and the value of diameter is contained in E41.

TOU (TOF) - Tool Expiry Declaration

The TOU command allows to declare the specified tool expired.

Syntax

(TOU,tool**)**

---

where:

tool	is the number of the tool number to be declared expired. It can be a local or a system variable

Example:

```
(TOU,5)    ;Declares tool 5 in the tool table expired
E1=10      ;
(TOU,10)   ;Declares tool 10 in the tool table expired
```

LOA - Table loading

The LOA command allows dual port loading of the disk-resident table.

Syntax

(LOA, *table name* [.*extension*] [,*par1*] [,*par2*]]**)**

where:

table name	it is the file name of the table to be dual port loaded.
extension	it is the extension of the file name and refers to the type of table to be loaded:

.TOL	table of tools
.OFS	table of adapters
.SPN	table of spindles (for the WOOD option)
.ORG	table of origins
.USR	table of user variable
.MAG	table of magazines

When there is no extension, the table referred to is the one pertaining to the Electronic Cam option (for description see relevant manual).

When tool and adapter have been associated in the loading of the tools table, also the relevant strings are loaded automatically.

The name of the table with its extension may also be defined by means of the variable SC.

par 1 It is a parameter having a different meaning according to the type of table to be loaded:

table of tools ⇒ *par 1* stands for the number of magazine the tools belong to and it is compulsory when magazines are being used.

table of origin ⇒ *par 1* par1 is the number of the process to which the axes relevant to the origins to be loaded belongs to.

par 2 It is considered only when loading the table tools together with the parameter par 1 and indicates whether the pocket values of the tools table need to be updated.

par 2 = 0	no pocket update
par 2 = 1	pocket update

When no parameter is available, 0 is chosen by default.

Characteristics:

The LOA command enables a dual port loading of the tables previously saved on file by TABLE EDITOR.

The names for the relevant extensions which identify the type of table are given by default and can be different if modified through the SET UP utility of TABLE EDITOR.

When using the magazines, the operation "pocket initialization" is to be accomplished by TABLE EDITOR. Such operation is necessary each time tables referred to magazines featuring different pockets are loaded or after reset of the dual port memory.

Examples:

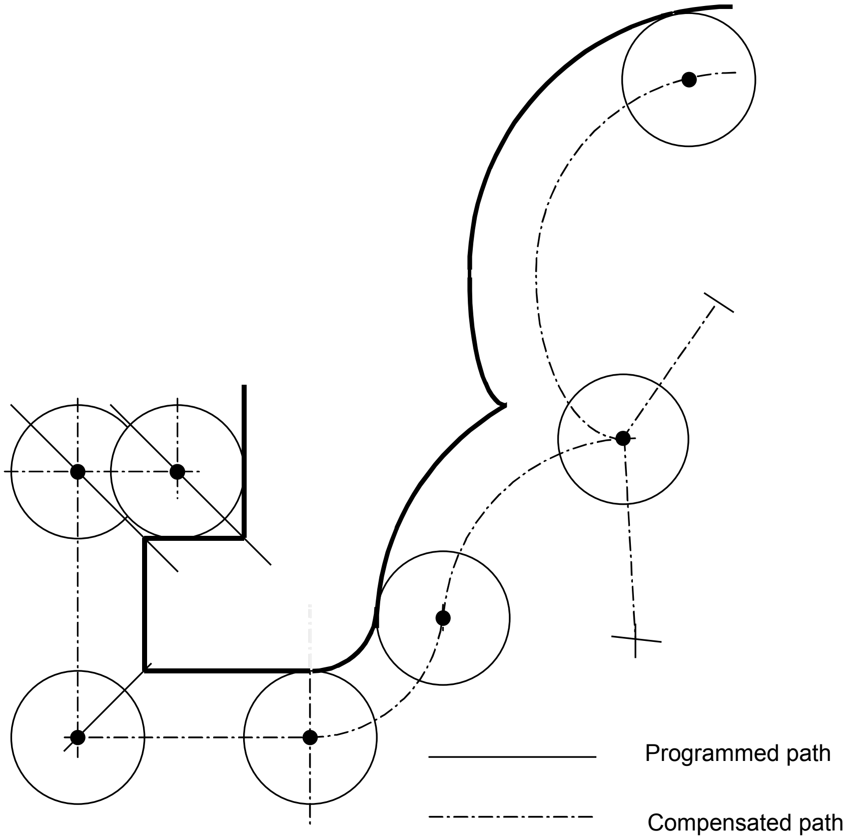
(LOA, NEW_TOOL.MAG)	loads the magazine table named NEW_TOOL.
(LOA, DRILL_T.TOL, 1, 1)	loads the DRILL_IT tools table relevant to magazine 1 and updates the pocket field of each tool.
(LOA, ORIG_1.ORG, 1)	loads the ORIG_1 table relevant to process 1.

END OF CHAPTER

CUTTER DIAMETER COMPENSATION

Cutter diameter compensation is based on the tool geometry. Since the typical tool section is a circle, the correction is applied to the tool diameter.

Cutter diameter compensation is perpendicular to a programmed profile consisting of straight line segments and circle arcs. The figure below shows how the tool is positioned when cutter diameter compensation is applied.



Tool Positioning for Cutter Diameter Compensation

When cutter diameter compensation is active, the tool is positioned on the intersection or tangency point of the two geometrical elements translated by the cutter radius.

G40 G41 G42 - Cutter Diameter Compensation

These G codes disable or enable cutter diameter compensation:

G40	Disables cutter diameter compensation
G41	Enables cutter diameter compensation when the tool travels left of the profile
G42	Enables cutter diameter compensation with the tool travels right of the profile

Syntax

G40 [*G-codes*] [*operands*]
G41 [*G-codes*] [*operands*]
G42 [*G-codes*] [*operands*]

where:

G-codes	Other G codes that are compatible with G41, G42 and G40 (see “Compatible G codes” table in Chapter 1).
operands	Any operand or code allowed in a G function block.

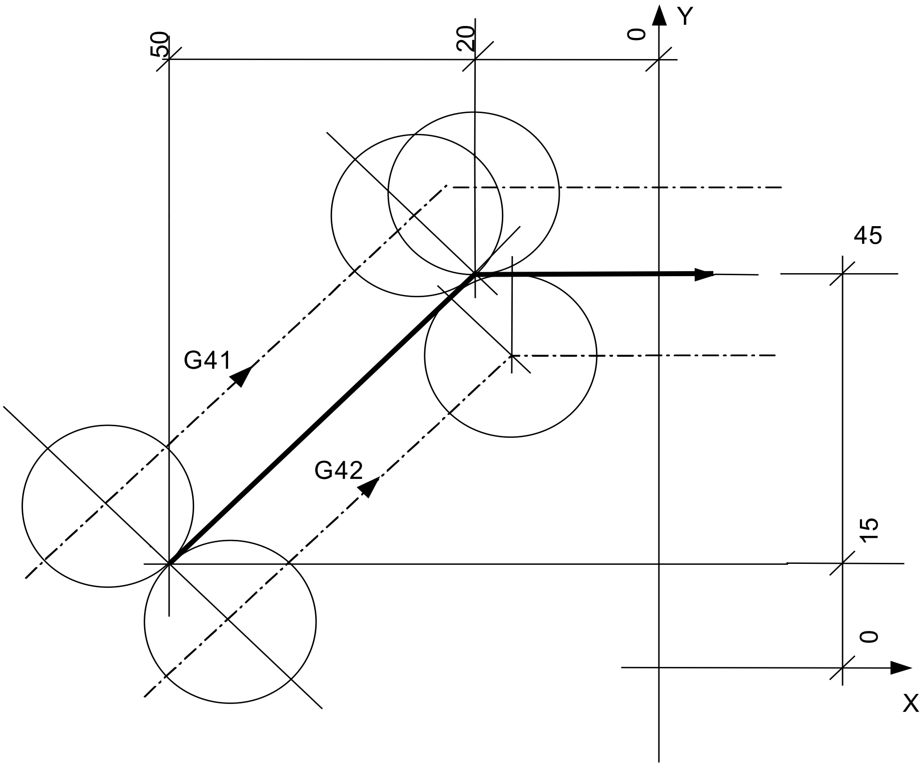
Enabling Cutter Diameter Compensation

Cutter diameter compensation is enabled by codes G41 or G42 . The movement to the first point in the profile must be linear (G00-G01).

On the first point in the profile, cutter diameter compensation is perpendicular to the first linear or circular move programmed after G41 or G42 on the active plane.

The examples below show how cutter diameter compensation is applied.

- **Linear move first in the profile**



Program:

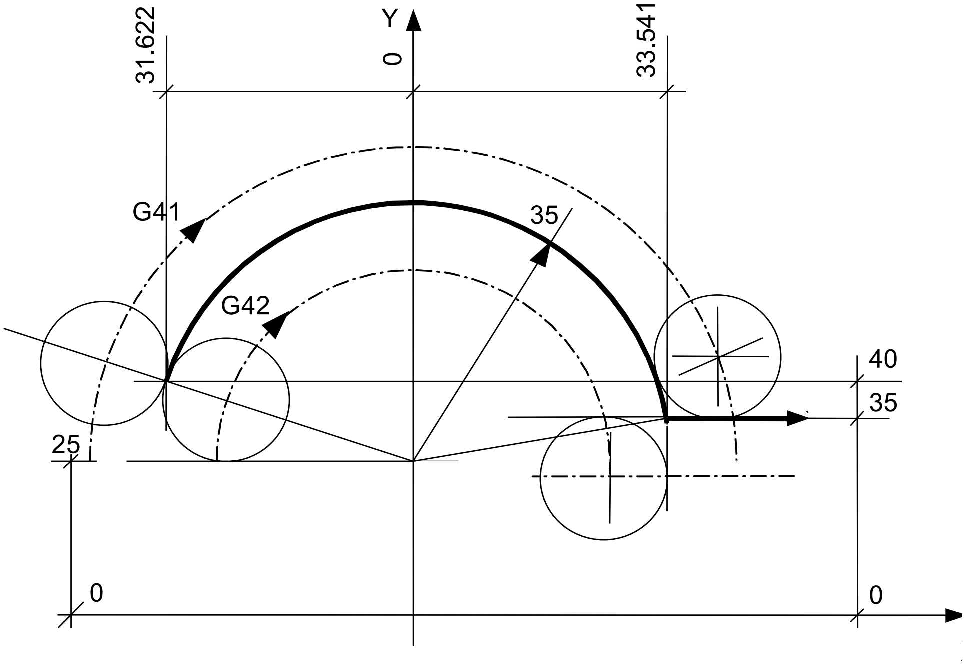
Cutter diameter compensation on the right of the profile:

```
G1 G42 X-50 Y15 F200
X-20 Y45
```

Cutter diameter compensation on the left of the profile:

```
G1 G41 X-50 Y15 F200
X-20 Y45
```

- **Circular move first in the profile**



Program:

Cutter diameter compensation on the right of the profile:

```
G1 G42 X-31.622 Y40 F200
G2 X33.541 Y35 I J25
G1 X.....
```

Cutter diameter compensation on the left of the profile:

```
G1 G41 X-31.622 Y40 F200
G2 X33.541 Y35 I J25
G1 X....
```

Notes on using cutter diameter compensation

Once enabled, cutter diameter compensation is applied to all the moves programmed at machining or rapid feedrate. After cutter diameter compensation has been enabled by G41 or G42, the following G functions cannot be programmed:

- G81 - G89 (canned cycles)
- G70 - G71 (mm/inch programming)
- G79 (programming referred to machine zero)
- G33 (threading)
- G72 - G73 - G74 (measuring cycles)
- G16 - G17 - G18 - G19 (change of interpolation plane)

When cutter diameter compensation is active, the control displays an error signal if:

- the programmed internal radius is shorter than the tool radius.
- the execution of a compensated linear move would reverse the tool direction with respect to the original profile.

Inside a profile compensated with G41 or G42 you can program up to six consecutive movements of axes that are not on the interpolation plane.

Tool path optimisation (TPO)

Tool path optimisation (TPO) may be enabled both from part program or with an MDI.

The function is programmed through the TPO system variable and may be customised to specific application requirements through the TPT system variable.

TPO permits to define two optimisation modes:

- automatic “reduction” of the tool path on corners between two linear or circular blocks
- infeed/exit tangent to profile (on a circle arc).

IMPORTANT TPO and TPT are discussed in detail later in this chapter.

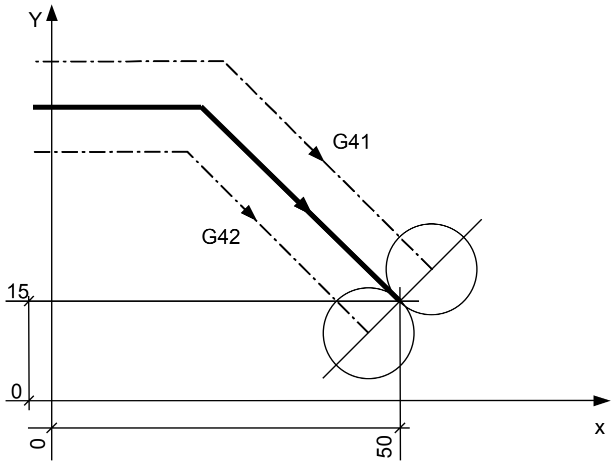
Disabling Cutter Diameter Compensation

Cutter diameter compensation is disabled by the G40 code. Then the cutter may exit from the part as follows:

- if TPO is not active (standard offset mode), the move programmed by G40 is still considered offset;
- if TPO is active, the move programmed by G40 is considered as the exit move from the offset.

These options are illustrated in the examples below.

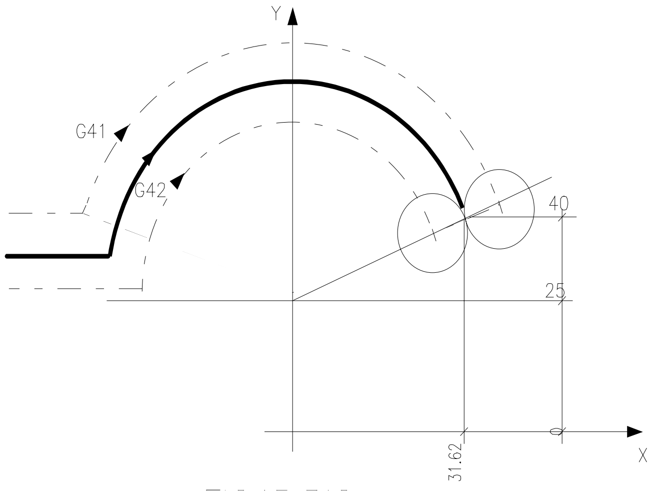
- ***The last move in the profile is linear (TPO=0)***



Program:

```
N88 G1 G40 X50 Y15
N100 G X.. Y..
```

- ***The last move in the profile is circular (TPO=0)***



Program:

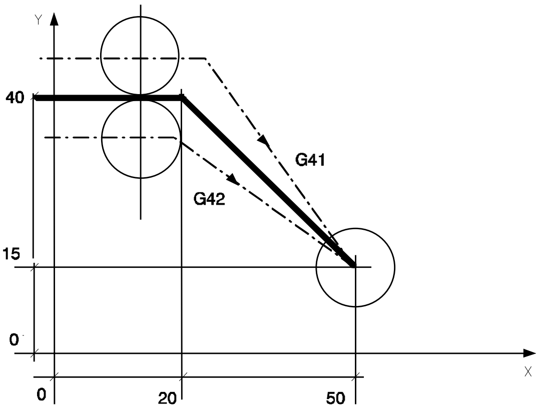
```
N99 G2 G40 X36.62 Y40 I J25
N100 G X.. Y..
```

Disabling Compensation with TPO active

With TPO active, the cutter diameter is always compensated through G40; the only difference with respect to the standard procedure is how the last block is executed.

These options are illustrated in the examples below.

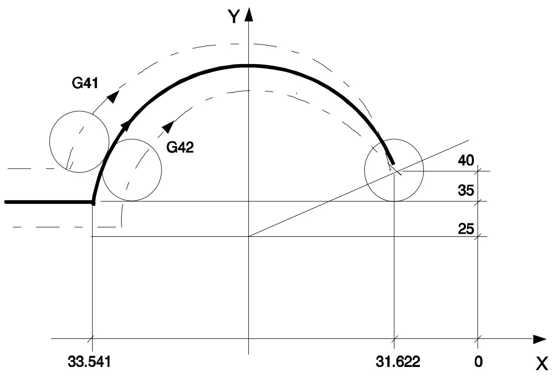
- ***The last move in the profile is linear (TPO active)***



Program:

```
N87   G1 X20 Y40
N88   G1 G40 X50 Y15
N100 G X.. Y..
```

- ***The last move in the profile is circular (TPO active)***



Program:

```
N98   G1X-33.541 Y35
N99   G2 G40 X36.62 Y40 I J25
N100 G X.. Y..
```

WARNING

If your release is earlier than 3.0 you must check whether G40 is programmed in your old part programs. When TPO is active, G40 exits from cutter diameter compensation somewhat differently.

If you program a block including only G40 (i.e. not associated to any final point), you will obtain the same result you would with standard diameter compensation (TPO=0).

Prior to writing a program read this section and the descriptions of TPO and TPT carefully.

TOOL DIAMETER COMPENSATION CHANGE

This section describes how changes of compensation type (G41 --> G42 and vice versa) are handled during offset profile machining. Compensation type changes may occur at the point of intersection of the programmed paths (with left/right or right/left compensation) or by the addition of a new movement block automatically by the system.

Type of compensation change (on the intersection or with an additional connection block) depends on type of the previous movements and of the subsequent movement. The different possibilities are discussed in the following pages:

- Linear/Linear with tangential movement blocks
- Linear/Linear with inversion of direction
- Linear/Linear with automatic generation of a new movement block
- Linear/Linear without automatic generation of a new movement block
- Linear/Circular - Circular/Linear
- Circular/Circular

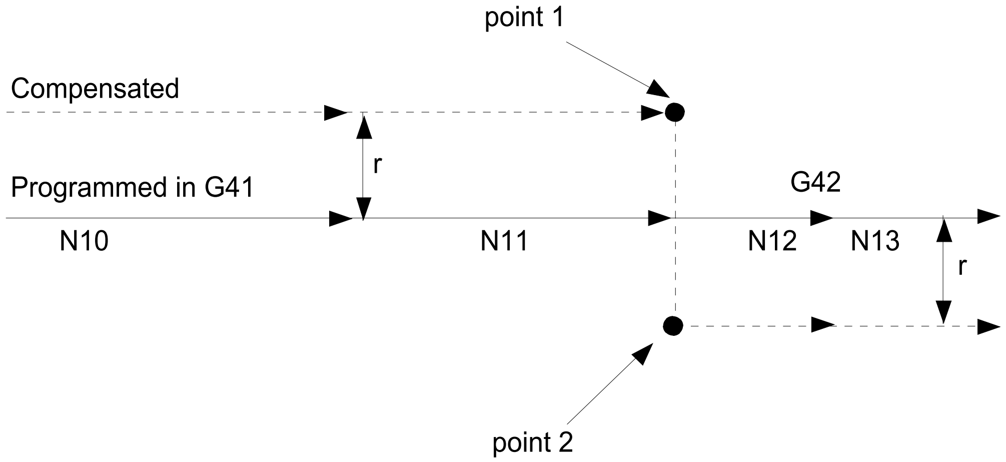
Linear/Linear tool path

The following figure illustrates the tool path when compensation changes from G41 to G42 during execution of two linear type movements.

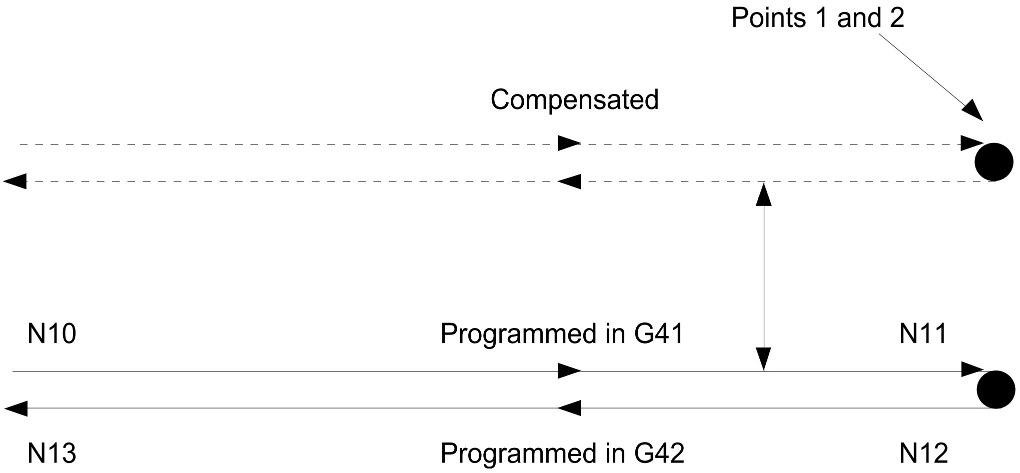
In changing from G41 to G42, the control generates two points, which shall be called *point 1* and *point 2*.

- *Point 1* is the final position of the tool before compensation type change.
- *Point 2* is the desired starting position for the first block using the changed compensation direction.

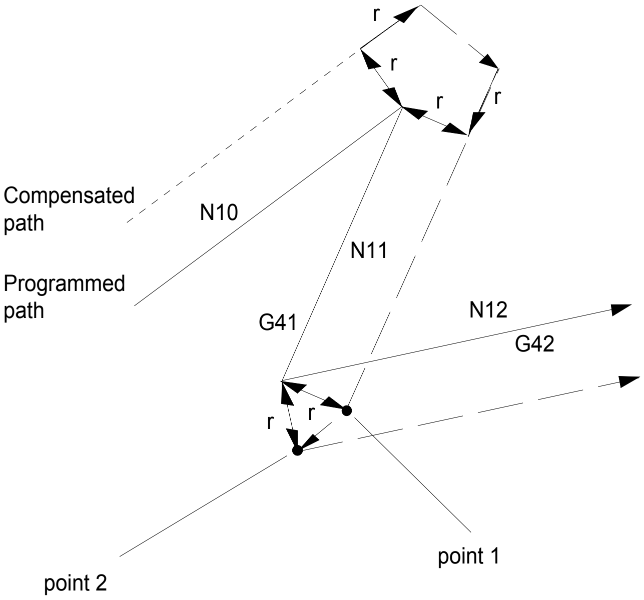
The control automatically generates the movement block connecting point 1 with point 2:



Linear/Linear change with tangential movement blocks



Linear/Linear change with inversion of direction

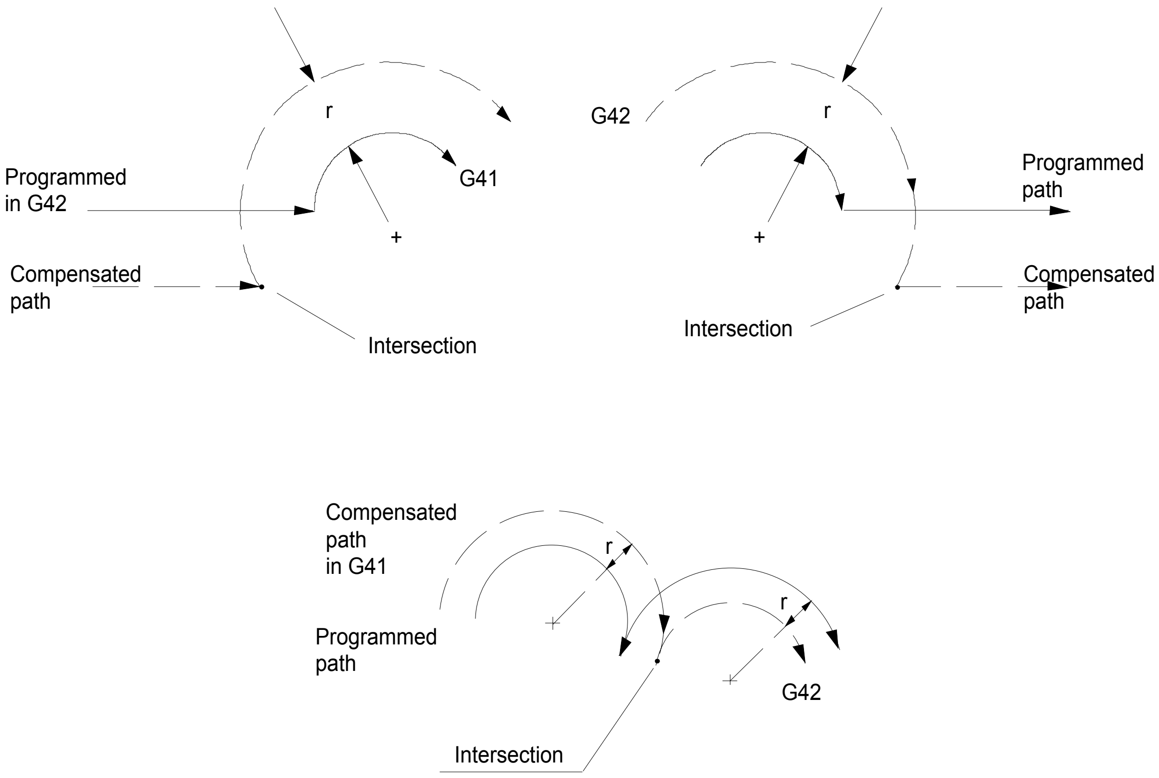


Linear/Linear change with generation of a new block

Linear/Circular, Circular/Linear, Circular/Circular tool paths

For each of the following types of tool path, in which a change of compensation direction occurs, the 10 Series system will try to find a point of intersection between the path programmed in G41 and that programmed in G42 (or vice versa).

If the 10 Series finds a point of intersection, it modifies the *final point* of the original compensated tool path whereas the *starting point* of the new compensated tool path will coincide with the intersection (see figure below).

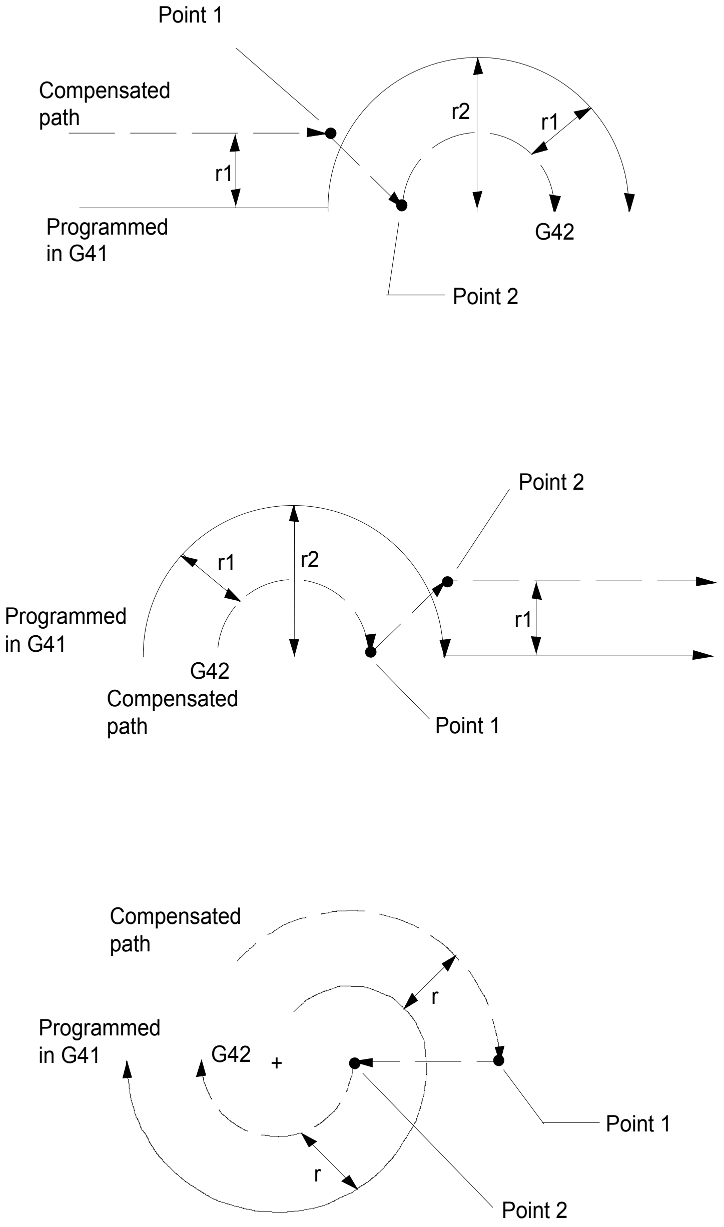


Change of compensation with intersection of current path

However, cases may arise in which there is no intersection between the tool paths; in these cases, when changing from G41 to G42 (or vice versa) the system behaves as illustrated in the figures that follow.

- *Point 1* is the final position of the tool before the change of compensation type
- *Point 2* is the desired position for start of the first block using the changed compensation direction.

The control automatically generates the movement block connecting point 1 with point 2:



Change of compensation without possibility of intersection between the tool paths

r - Radiuses in Compensated Profiles

When machining convex profiles, you may want a circular radius between geometric elements.

Syntax

---

r value

---

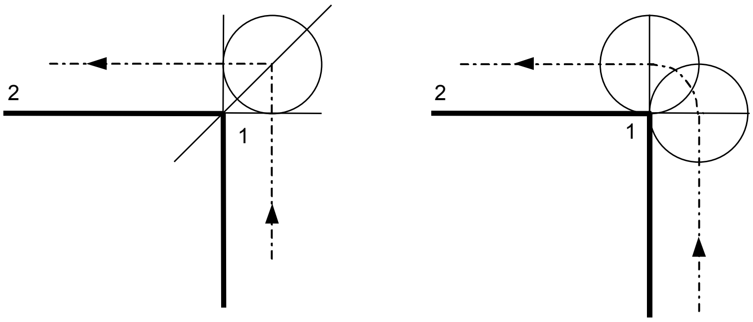
where:

value The radius to be programmed. For clockwise moves program a negative radius; for counter clockwise moves program a positive value.

IMPORTANT

Programming r0 (radius equal to zero) causes the tool centre to follow a circular arc whose centre is on the profile corner.

Example:



Without radius

1) N20 G1 X100 Y100
2) N21 X-100

With radius

1) N20 G1 X100 Y100
N21 r0
2) N22 X-100

b - Bevels in Compensated Profiles

By programming a value without a sign after the b address you can insert a bevel rather than a radius between two linear or circular motion blocks that generate intersecting paths.

Syntax

---

b value

---

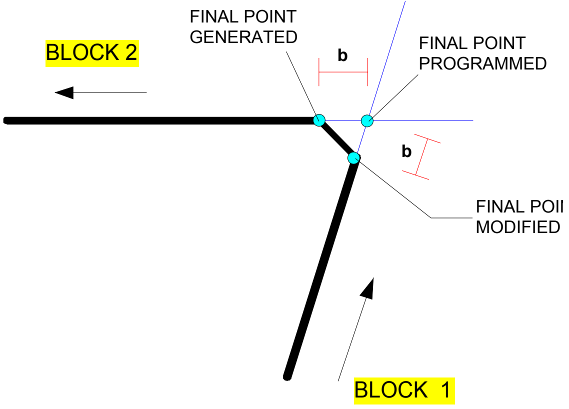
where:

value It is the bevel length measured from the intersection point.

This value **b** may be interpreted as follows:

- **Bevel between two linear profiles: *b*** is the distance from the generated final point to the theoretical intersection point between the extended profile segments.
- **Bevel between a linear and circular profile: *b*** is the distance from the theoretical intersection point between the extension of the linear profile to the tangent of the circular profile
- **Bevel between two circular profiles: *b*** is the distance to the theoretical intersection point between the extensions of the tangents to the circular profiles

Examples:

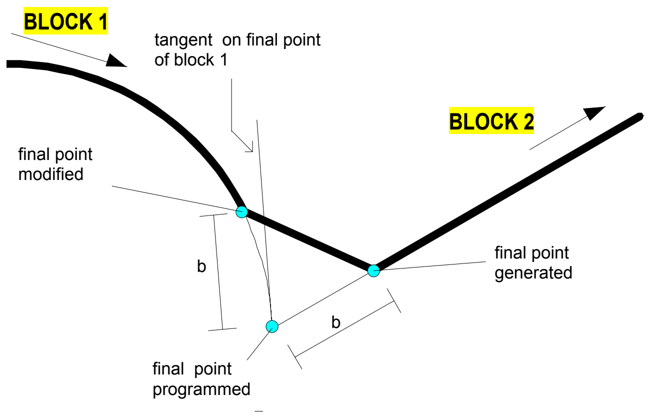


Bevel between two linear blocks

Programming example:

```
          .
          .
N10 G1 X10 Y100     ;block 1
N11 b3
N12 X-100           ;block 2
          .
          .
```

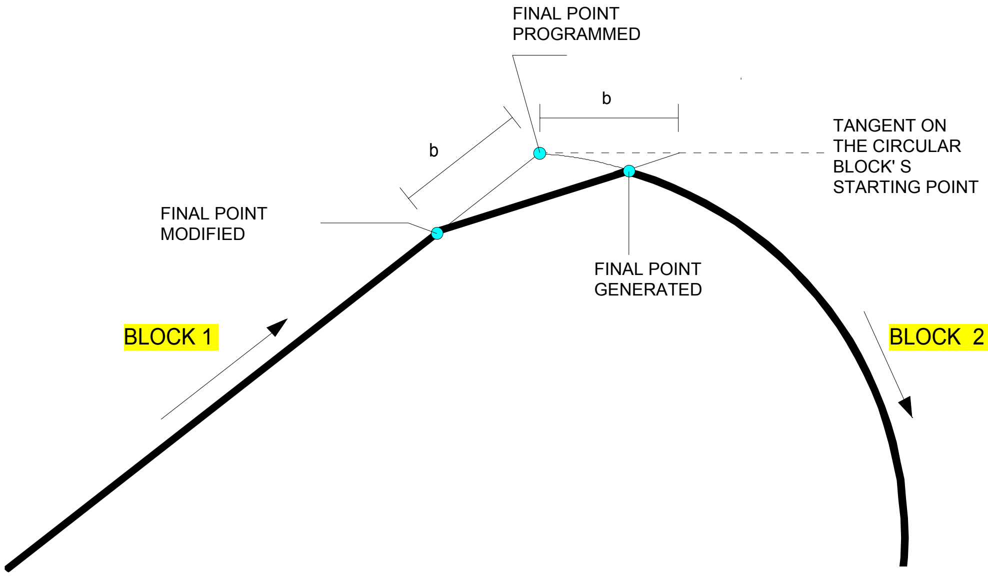
Bevel between circular and linear motion blocks



Programming example:

```
.
N10 GXY
N11 G42
N12 G1X10Y70F1000
N14 G2X70Y20R50      ;block 1
N16 b10
N18 G1X150Y90        ;block 2
N20 G40
.
.
.
```

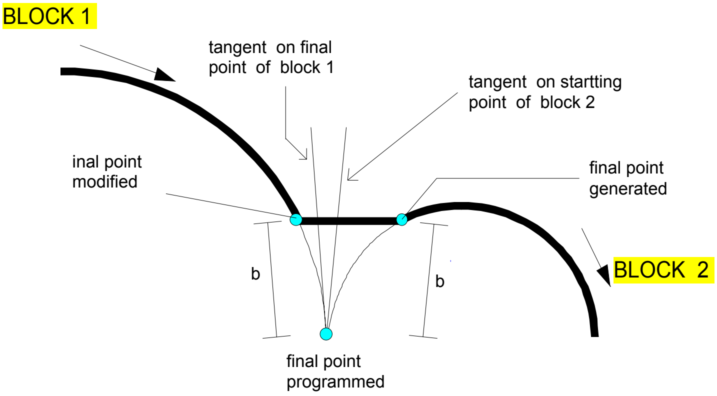
Bevel between linear and circular motion blocks



Programming example:

```
        .
N20     G42
N22     GXY
N24     G1X40Y40F1000       ;block 1
N26     b10
N28     G2X80Y5R70          ;block 2
N30     G40
        .
        .
```

Bevel between two circular motion blocks



Programming example

```
        .
        .
N10     G42
N20     GX10Y60
N30     G2X50Y40R50F1000    ;block 1
N40     b5
N50     G2X100Y5050R30      ;block 2
        .
        .
```

TPO - Path optimisation on bevels with G41/G42

TPO (Tool Path Optimisation) allows to optimise the tool path when G41 or G42 are active.

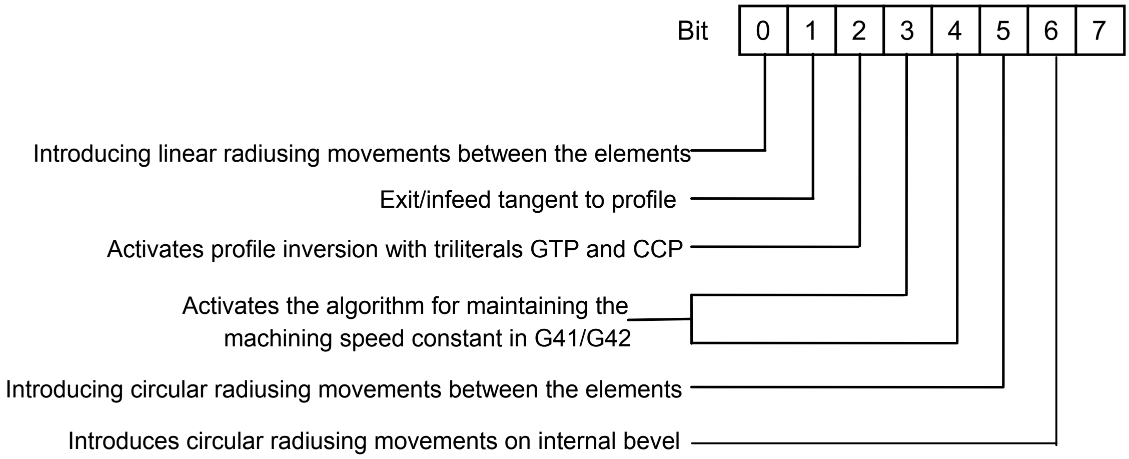
The algorithm automatically introduces circular interpolations at profile start and end, and linear or circular radiuses on the profile bevels.

Syntax

TPO = ***n***

where:

n	Optimisation mode. Allowed values are: The value to be specified for *n* in the instruction is obtained from the sum of the decimal weights corresponding to each of the features desired. If *n* = 0 the optimisation algorithm is disabled.


Bit
0
1
2
3
4
5
6
7
Introducing linear radiusing movements between the elements
Exit/infeed tangent to profile
Activates profile inversion with triliterals GTP and CCP
Activates the algorithm for maintaining the machining speed constant in G41/G42
Introducing circular radiusing movements between the elements
Introduces circular radiusing movements on internal bevel

Characteristics:

A block programming the TPO=n instruction activates the tool optimisation algorithm.

This algorithm can automatically introduce movements at the beginning/at the end of the profile and it can optimise the path on corners. Depending on the profile, the algorithm automatically introduces from 1 to 3 optimisation moves.

IMPORTANT

TPO is ignored in a GTL program.

IMPORTANT

At power up the system is enabled with the TPO mode configured in AMP. This optimisation mode may be altered from the program.
RESET restores the mode configured in AMP.

IMPORTANT

The activation of bit 5 and/or of bit 6 works only if bit 0 is enabled too, i.e. when you want to introduce a circular radius in lieu of a linear one, set **TPO = 33**.

If you want to introduce a circular radius on the inner corners of a profile, set **TPO = 65**

- **TPO=1, TPO=33 and TPO=65 modes**

Mode TPO=1 activates the tool path optimisation algorithm by introducing linear radiusing movements between the profile elements as a function of the angular deviation between the elements themselves.

Furthermore, if bit 5 has been set too (TPO=33), then, provided that the conditions for corner optimisation are met, through the introduction of linear movements a circular radius is introduced, having its centre at the intersection of the two non-compensated elements and a radius corresponding to tool radius.

If bit 6 is active too (TPO= 1+32+64=97), the above considerations are applied to the inner corners.

The cases of application of the algorithm are as follows:

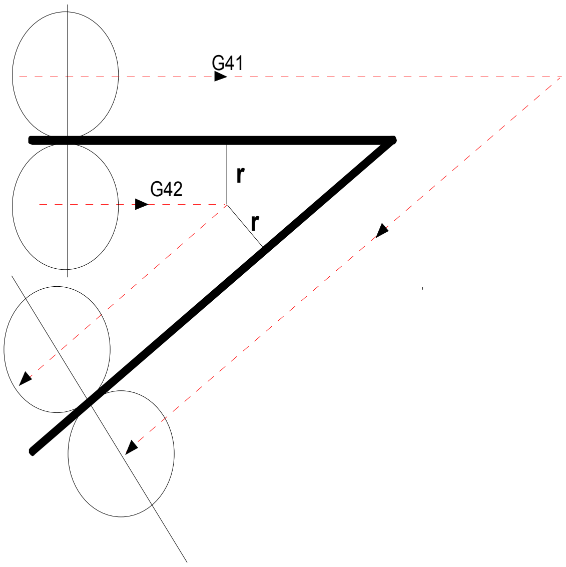
A) Right angle-right angle with angle beta of deviation between 0° and TPA ($0° < \beta <$ TPA°), with variableTPA, which is set by default on 90°

B) Circle-right angle, right angle-circle, circle-circle with angle of deviation between 0° and 180° ($0° < \beta < 180°$).

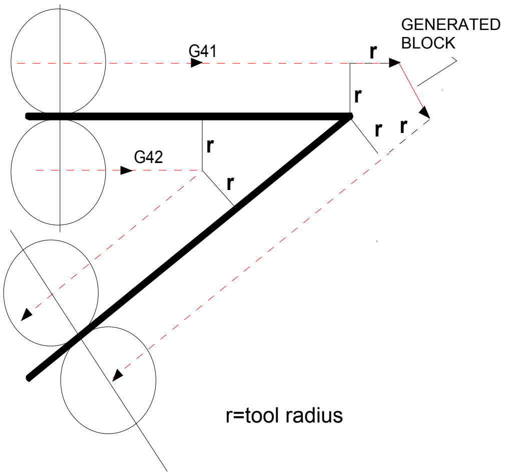
If either a bevel (b) or a radius (r) has been programmed between the elements described in A) and B), the algorithm is not applied.

Example:

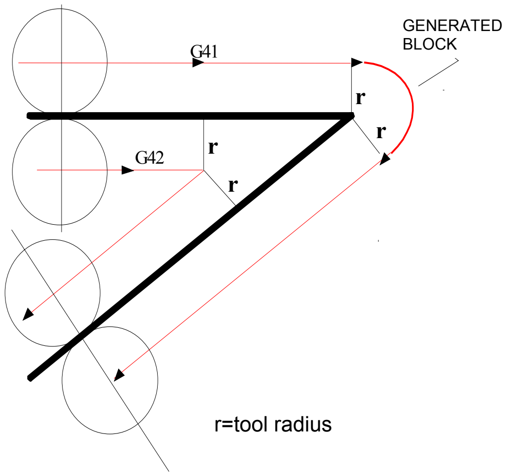
In this example, β is smaller than 90°



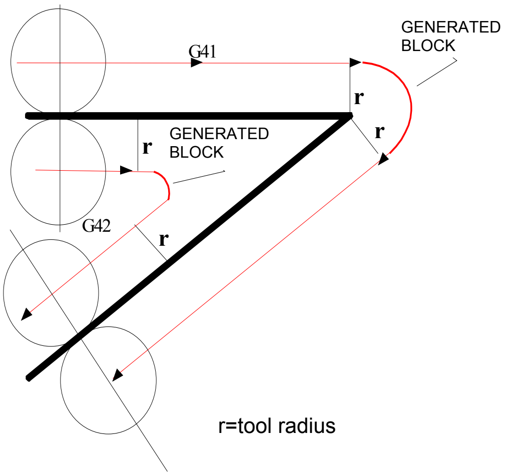
Without optimization (TPO =0)



With optimization (TPO=1)



With optimization (TPO=33)



With optimization (TPO=65)

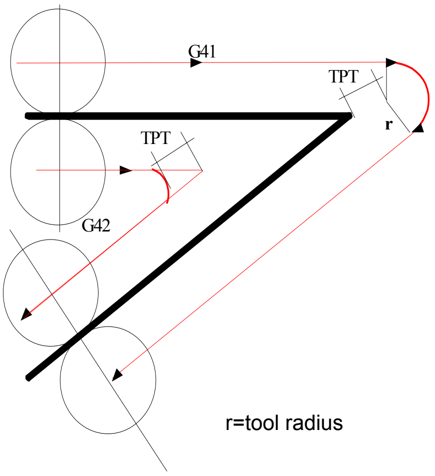
IMPORTANT

Note that in the above examples the TPT threshold has been ignored, as if TPT=0.

The figure below illustrates the strategy of TPO when TPT is not zero, i.e., a threshold deviation from the corner is programmed:

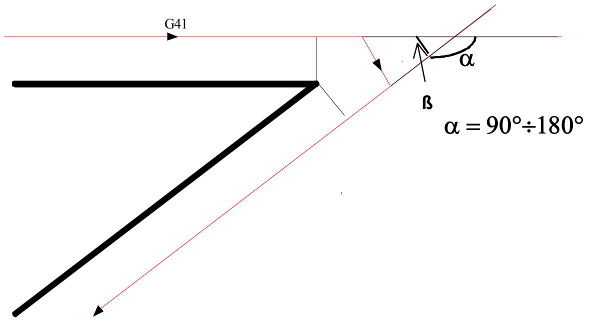
- in the case of an **outer corner**, the afore-mentioned circular radius will have its centre translated according to the same calculation criteria as are used for linear radiuses, and the radius will be reduced accordingly.

- in the case of an **inner corner**, the radius of the circular radius is calculated so as to have the tool path deviate from TPT by the amount that would be obtained without the introduction of the radius

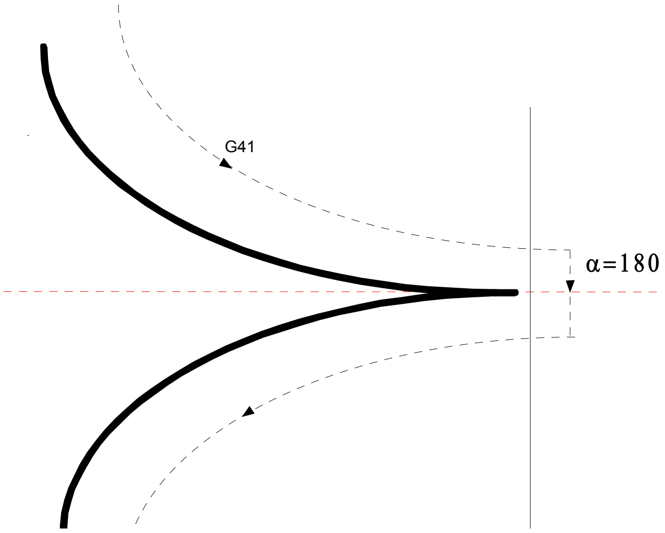


Examples of profile optimisation with TPO=1

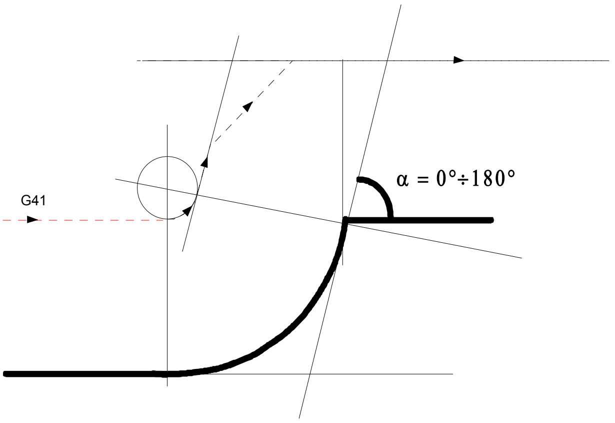
In these profiles the angle deviation requires an optimisation algorithm.



Angles from 0° to 90°
formed by two straight lines



180° angle with circle-circle
intersection

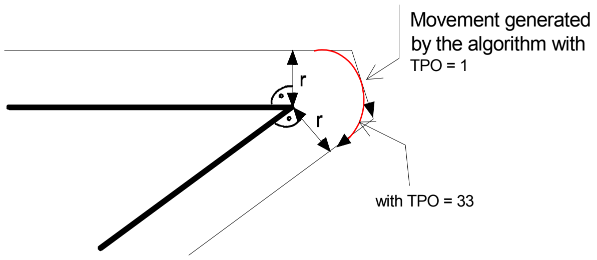


Angle from 0° to 180°
formed by a circle-straight
line intersection

Examples of profiles where algorithm introduces optimization moves

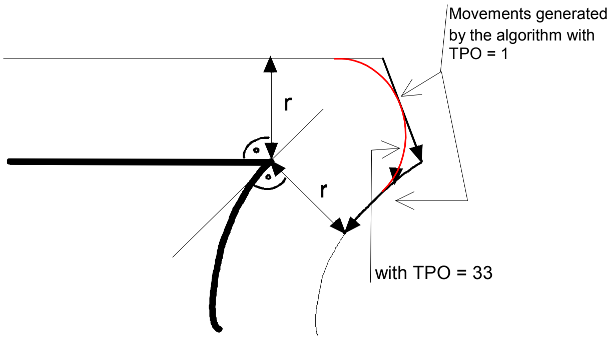
In these examples the algorithm recalculates the profile and introduces from 1 to 3 optimisation moves between bevels.

1) Line - line



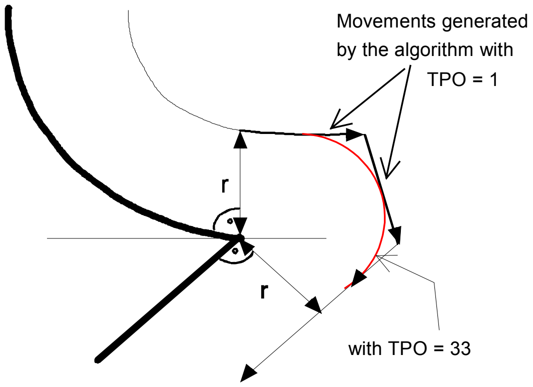
The algorithm generates 1 move

2) Line - circle



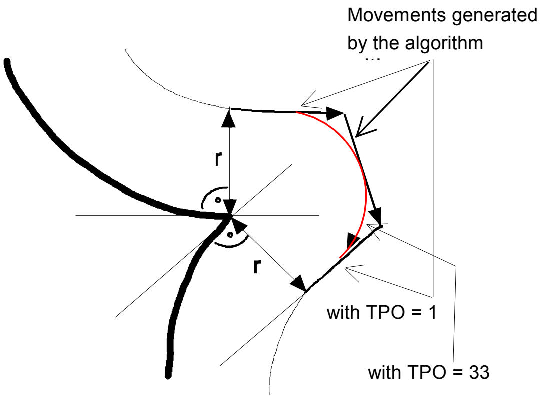
The algorithm generates 2 moves

3) Circle - line



The algorithm generates 2 moves

4) Circle - circle



The algorithm generates 3 moves

- **TPO=2 mode**

TPO=2 enables an infeed/exit algorithm that keeps the tool tangent to the profile by introducing circular elements at profile start and end.

At profile infeed the algorithm generates a circle between the first point of the offset profile (P1) and the preceding point (P0). The first point is the one programmed in the G41 or G42 block.

Example:

```
.....
X80 Y70               ;P0
G41 X100 Y100         ;P1
X140                  ;P2
.....                 ;etc.
…..
```

At profile exit the algorithm generates a circle between the last point of the offset profile (P99) and the last point of the exit element in the profile (P100). The exit element is programmed in the G40 block.

Example:

```
.....
G41 X100 Y100         ;P1
…..
.....
.....
.....
X170 Y160             ;P99
G40 X180 Y185         ;P100
.....                 ;etc.
.....
…..
```

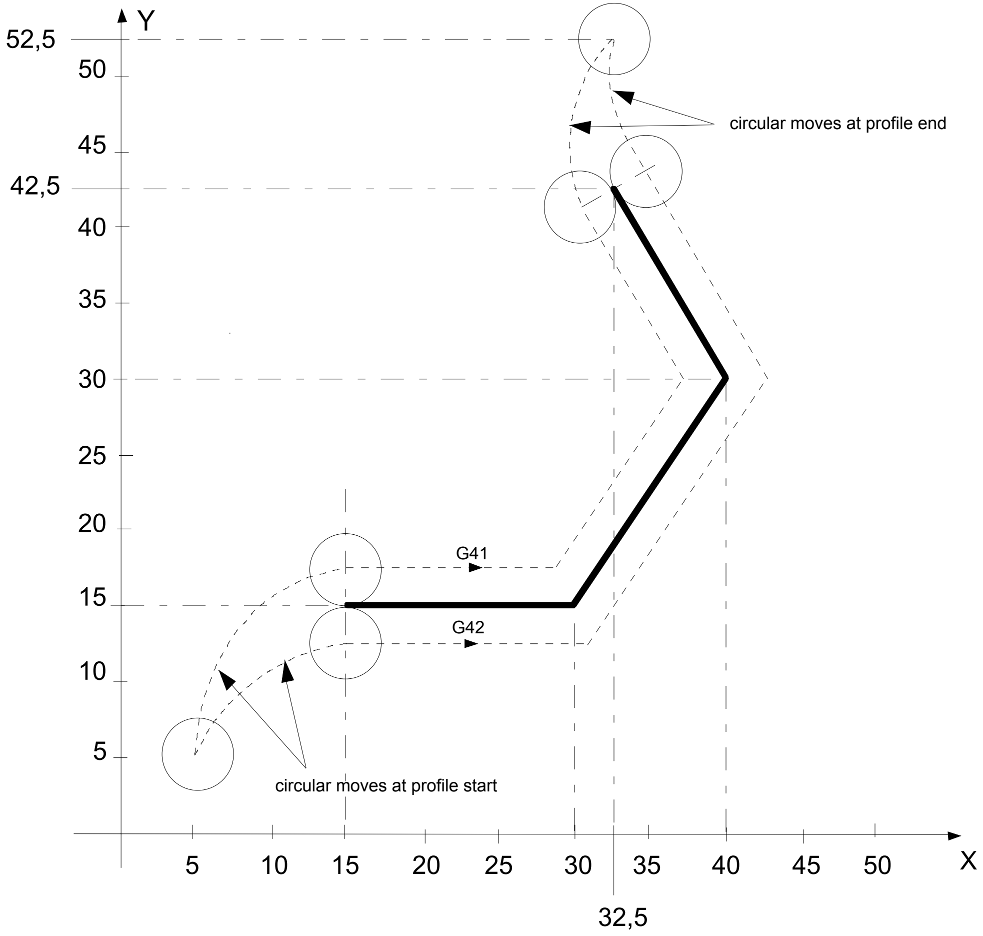
- **TPO=3 mode**

TPO=3 simultaneously enables TPO=1 and TPO=2 algorithms, which allows to introduce both radiuses between elements and circular moves at profile start and end.

The rules defined for TPO=1 and TPO=2 also apply to TPO=3.

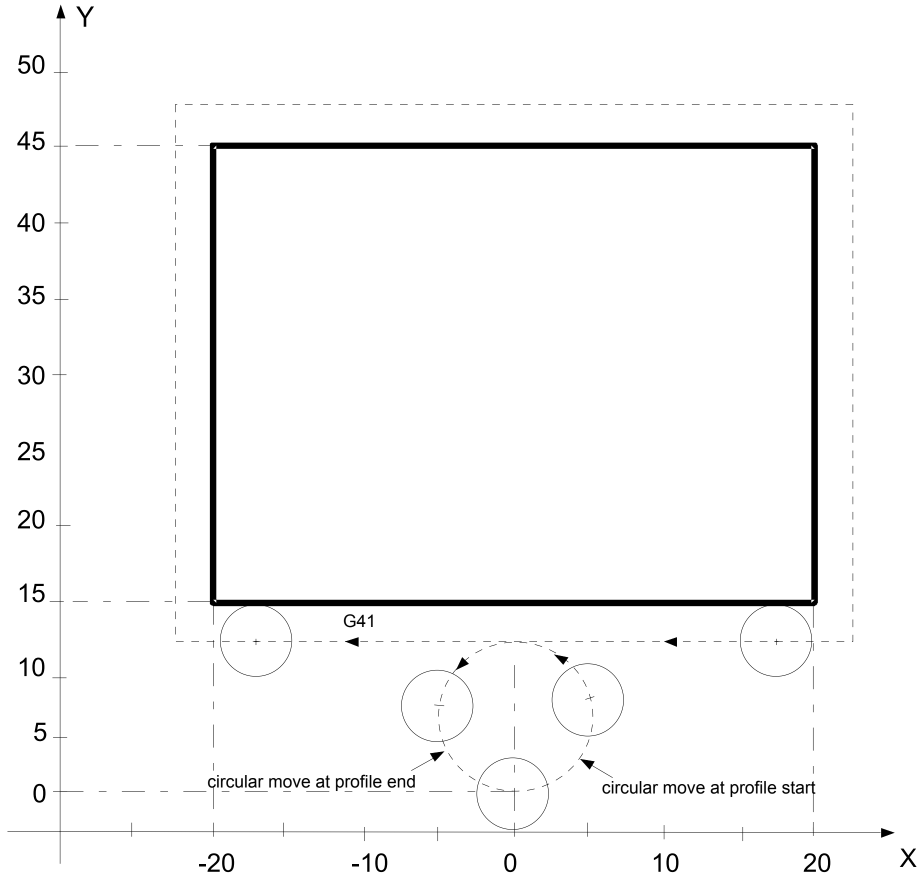
Examples of TPO=2 mode

Example 1:


Y
52,5
50
45
42,5
40
35
30
25
20
15
10
5
circular moves at profile end
G41
G42
circular moves at profile start
5
10
15
20
25
30
35
40
45
50
X
32,5

```
N1      S1000 M3 T1.1M6           ;tool radius = 2.5
N2      X5 Y5
N3      G1 G42 X15 Y15 F500       ;first point in the profile
N4      X30
N5      X40 Y30
N6      X32.5 Y42.5               ;last point in the profile
N7      G40 X32.5 Y52.5
N8      GX100
```

Example 2:



```
N1     S1000 M3 T1.1 M6          ;tool radius = 2.5
N2     X0 Y0
N3     G1 G41 Y15 F800           ;first point in the profile
N4     X -20
N5     Y45
N6     X40
N7     Y15
N8     X0                        ;last point in the profile
N9     G40 Y0
```

- **TPO=4 mode**

The TPO=4 mode activates the profile inversion algorithm.

This feature permits machining of a profile starting from the last movement, and finishing on the first (reverse machining).

IMPORTANT

- This feature in managed in connection with commands GTP and CCP only.
- The algorithm also provides for inversion of any offset modes, so as to maintain position of the inside/outside tool as required.
- For further details, see the pages dealing with the triliterals GTP and CCP (page 4-37 and 4-40).

- **TPO=8 and TPO=16 modes**

TPO=8 and TPO=16 modes activate the algorithm that keeps the machining speed in G41/G42 constant. In particular, the speed of contact between the tool and the part is kept constant, varying the feed rate with reference to the centre of the tool.

This variation is a function of the tool radius and is only applied to circular movements. It will produce an increase or decrease in the feed rate with reference to the centre of the tool depending on whether the radius of the circle moved by the centre of the tool is greater or less than the radius of the programmed circle. The feed rate is increased by setting bit 3 of variable TPO (TPO=8) while it is decreased by setting bit 4 (TPO=16).

Bits 3 and 4 of TPO may be set together and even at the same time as all the other bits of the same variable to enable the corresponding features.

- **TPO=32 mode and 64 mode**

Is the same as having programmed TPO = 0.
This is so because the activation of external and inner bevel optimisation through the introduction of a circular radius requires the prior programming of TPO = 1. If bit 0 = 0 and bit 5 and/or bit 6 = 1 (TPO=32 and/or TPO=64), the optimisation algorithm by means of circular radiuses is not enabled.

TPA – Threshold angle for TPO

System variable that consists of the threshold angle within which two linear elements can be radiused in tool compensation mode (G41/G42).

Syntax

TPA = *value*

---

where:

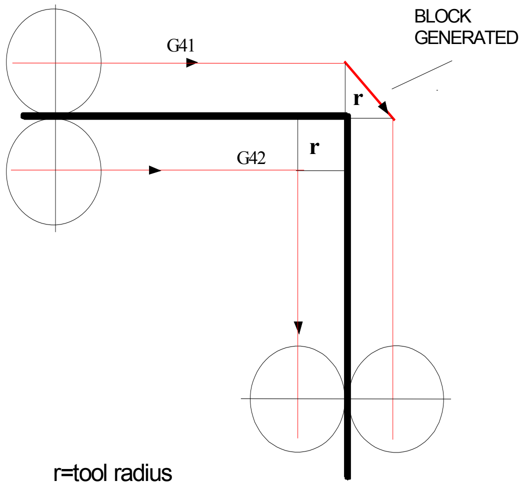
value	Threshold angle expressed in degrees. If the angle between two adjacent elements is smaller than the value of TPA, a circular or linear radius is introduced in accordance with the values set in TPO. The default value of TPA is = 90°; this value can also be configured in AMP. If the value has been changed by the program, RESET restores the default value.

Characteristics

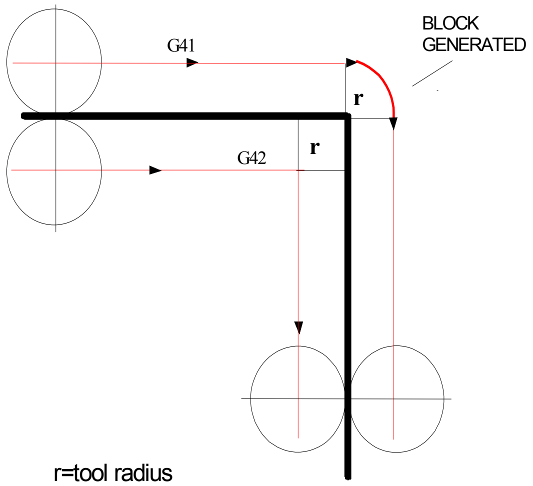
This variable has been introduced to be able to configure the threshold angle used by the Tool Path Optimisation (TPO) algorithm to radius two linear elements in tool compensation mode: so far, this angle was fixed at 90°. If at least one element is circular the optimisation algorithm remains unvaried, i.e., a radius (either linear or circular, depending on the value of TPO) is introduced regardless of the angle between the two elements.

Example:

With TPA > 90° and TPO = 1



With TPA > 90° and TPO = 33



TPT - Tool Path Threshold

This instruction specifies a threshold for bevels during tool path optimisation when TPO=1.

Syntax

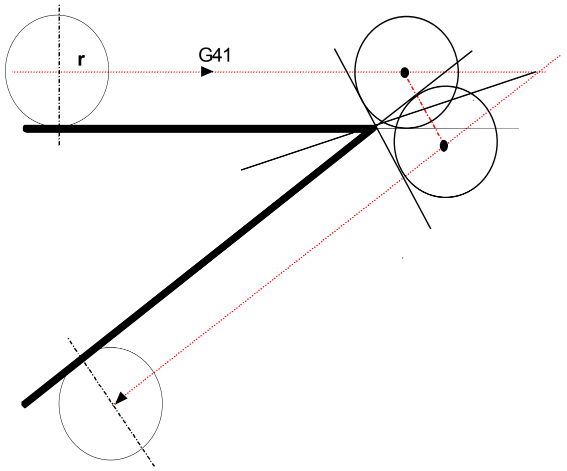
TPT = value

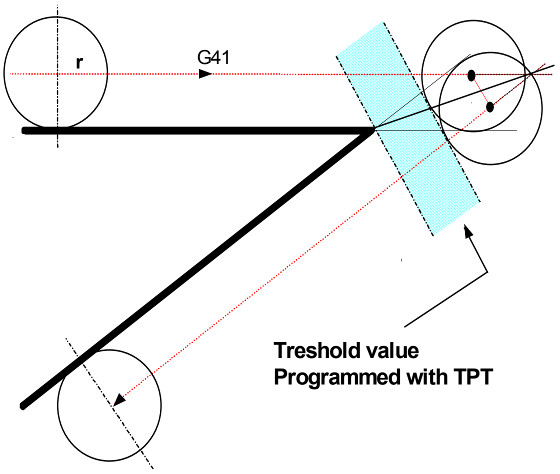
---

where:

value	Threshold expressed in the default unit of measure (mm/inch). It represents the distance between the tool cutter and the bevel generated by the programmed profile. It can also be configured in AMP.

Example:





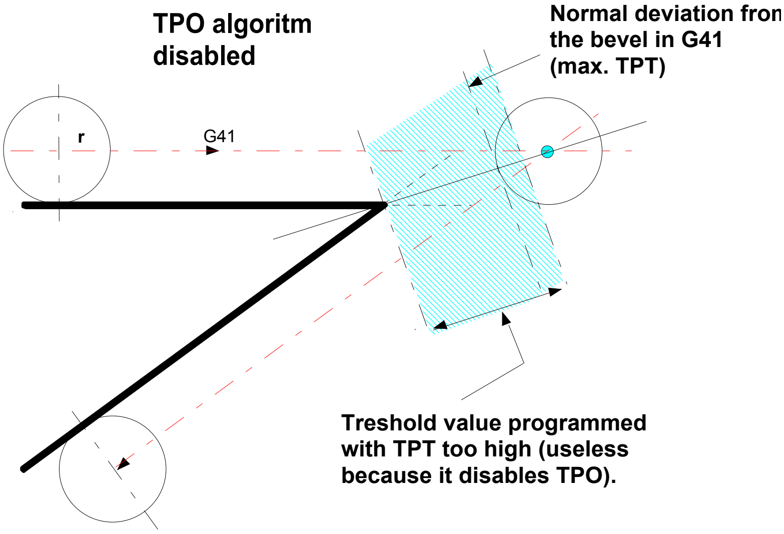
**Treshold value
Programmed with TPT**

Characteristics:

The threshold programmed with TPT will be ignored by the system if it takes the mill centre beyond the theoretical point obtained through standard tool diameter compensation (G41/G42 without TPO programming).

In this case the TPO algorithm will be temporarily disabled, G41/G42 will be applied without TPO and there will be no error signal.

Example:


TPO algoritm disabled
Normal deviation fron the bevel in G41 (max. TPT)
r
G41
Treshold value programmed with TPT too high (useless because it disables TPO).

u v w - Paraxial Compensation

When compensation factors u,v,w are programmed in a block, the axes position to a point whose coordinates are equal to the programmed coordinates plus the product of the cutter radius by the compensation factor (u, v, w):

X position = programmed X + (cutter radius * u)

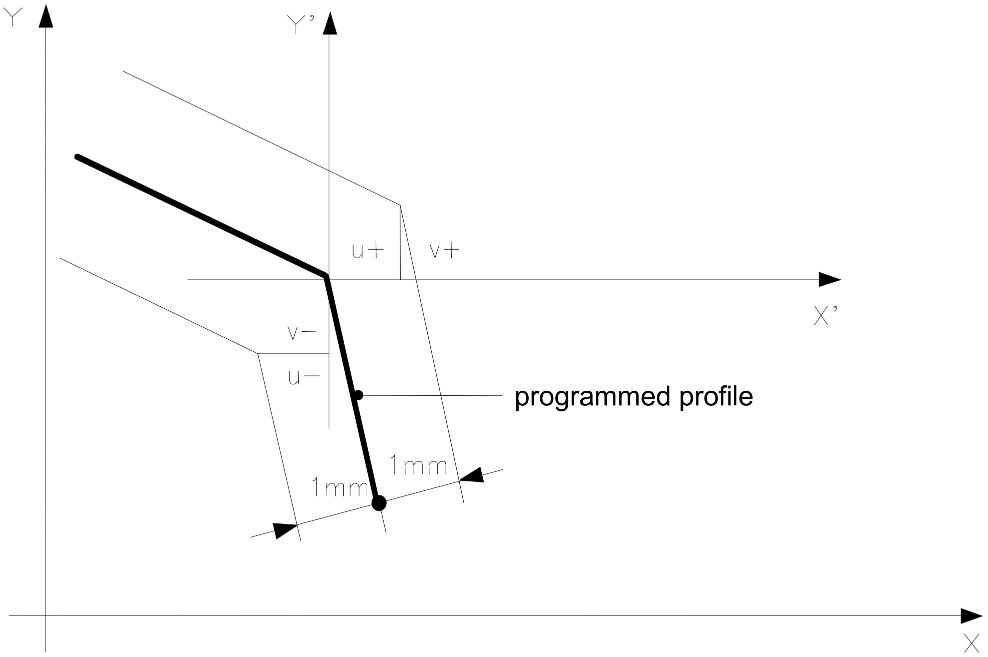
Y position = programmed Y + (cutter radius * v)

Z position = programmed Z + (cutter radius * w)

These compensation factors are used both for extremely simplified profiles (contouring parallel to axes) and three-dimensional milling surfaces. You cannot use the u,v,w factors when the control is in cutter diameter compensation mode (G41-G42).

When factors u,v,w are negative, they must be followed by a minus sign. If they are positive, the plus sign can be omitted.

In order to determine the value and the sign you must consider paraxial factors u,v,w respectively as the X, Y, Z coordinates of the profile vertex corrected by a unit. X, Y and Z are referred to a system of cartesian axes that are parallel to the axes of a machine whose origin is represented by the point to be compensated.



Determining the sign of u, v, w compensation factors

The unit vector represents a cutter radius offset that is one unit long; the control, the increments of translation are the product of the u,v,w values multiplied by the cutter radius.

You can compensate profiles that consist of:

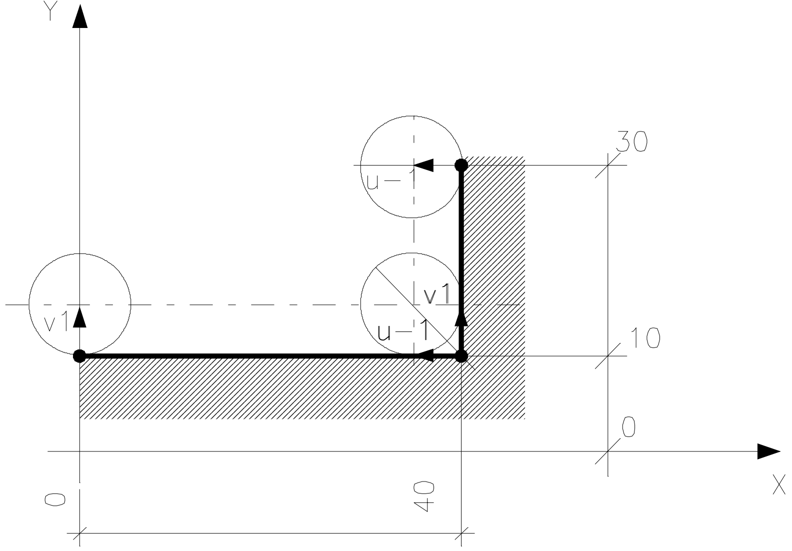
- straight line segments parallel to the axes or forming an angle with the axes
- straight line segments and arcs tangent to the straight lines
- tangent arcs (provided they are still tangent to one another after they have been translated on a parallel path).

The u,v,w factors are only valid in the block in which they are programmed. They are used by the control only if they are associated to the corresponding coordinates:

- u for the 1st configured axis (typically X)
- v for the 2nd configured axis (typically Y)
- w to the 3rd configured axis (typically Z).

Examples of compensation factor applications u, v, w

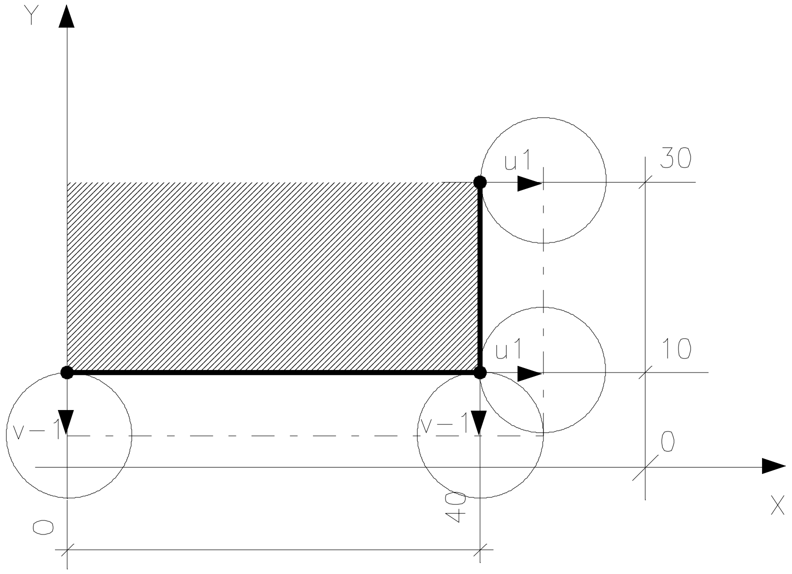
Example1:



Program:

```
N5 T1.01 M6 S. . F. .
N6 G X Y30
N7 G1 Y10 v1
N8 X40 u-1
N9 Y30
N10 G X. . Y. .
```

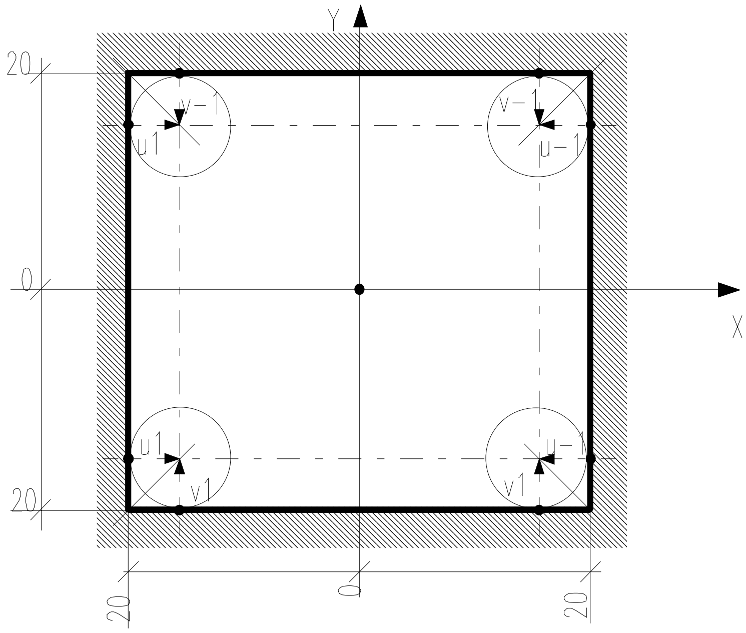
Example 2:



Program:

```
N13 G X Y
N14 Y10 v-1
N15 X40 u1
N16 Y30
N17 G X. . Y. .
```

Example 3:

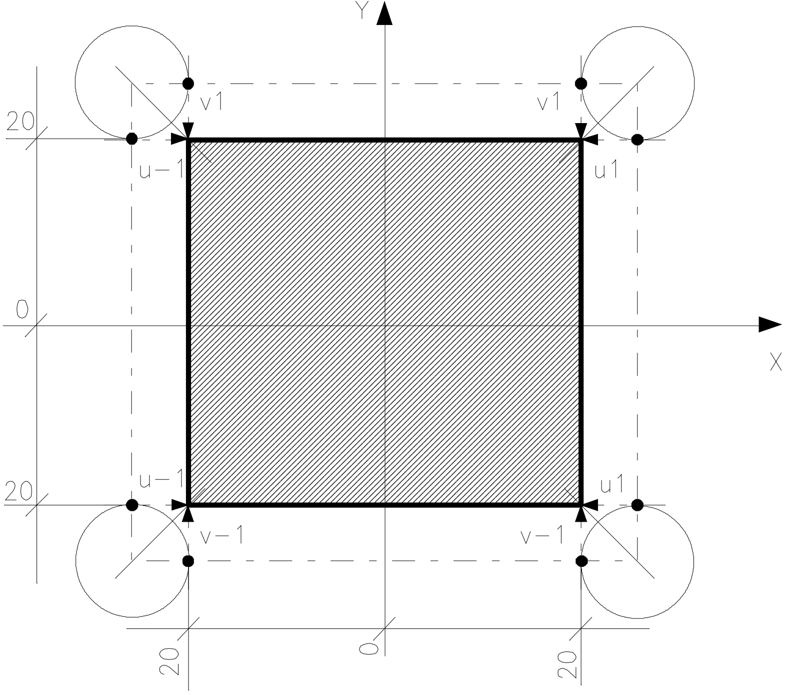


Program:

```
N13 G X Y
N14 G1 Z-10
N15 X-20 Y-20 u1 v1
N16 X20 u-1
N17 Y20 v-1
N18 X-20 u1
N19 Y-20 v1
N20 G X Y
```

FIG57.PIC

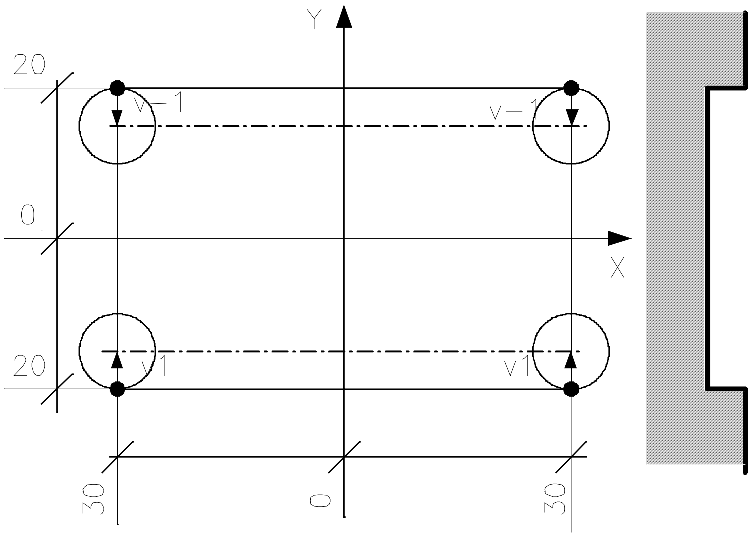
Example 4:



Program:

```
N13 G X-35 Y-35
N14 Z-10
N15 G1 X-20 Y-20 u-1 v-1
N16 X20 u1
N17 Y20 v1
N18 X-20 u-1
N19 Y-20 v-1
N20 GZ
```

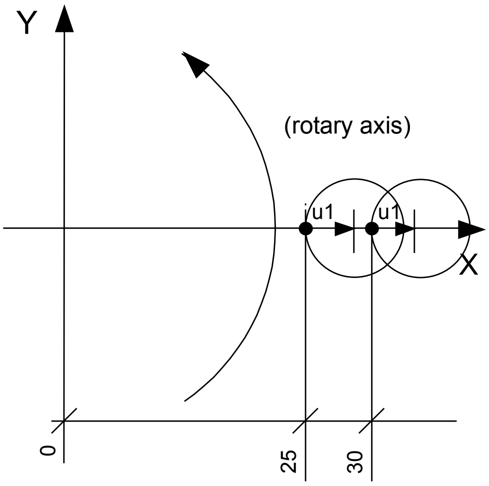
Example 5:



Program:

```
N12 . . .
N13 G X-30 Y
N14 G1 Y20 v-1
N15 X30
N16 Y-20 v1
N17 X-30
N18 . . .
```

Example 6:



Program:

```
N12 G X40 Y B0
N13 G1 X30 u1
N14 X25 B360 u1
N15 G X40
```

UVW – Definition of axes for paraxial compensation

It is possible to change the axes on which paraxial compensation is performed.

As a rule, paraxial compensation is applied, within each process, on the first three axes configured (the first three to appear on the screen). With triliteral (**UVW**) you can change the axes on which paraxial compensation is performed.

Syntax

(**UVW**, axis1 axis2 [axis3])

where:

axis1…axis3	are the names of the axes to be associated with the **uvw** paraxial compensation: axis 1 is associated with parameter **u**, axis 2 with parameter **v**, and axis 3 with parameter **w**..

Characteristics:

If only two axes are specified in the command, the paraxial compensation will be activated only for parameters **uv**, and the third parameter (**w**) will be ignored.

Once defined, the association between the axes and the uvw parameter remains active until the next RESET; then the default association (the first three axes configured) is restored. A GTA command and the activation of a virtualisation (UPR, UVA, UVP…) will also restore the default association, unless all the axes (construed as axis ID) on which the uvw parameters are active remain present even after the execution of the command.

Example:

```
N10 (UVW,XYZ)                   Activates uvw on X Y and Z respectively
N20 h1
N20 X10u1Y20Z40
N40 (UPR,XYZ,PQD,20,20,0)       Activates UPR, the uvw parameters are activated on the
                                first three axes configured

N50 (DAN,XP,YQ,ZD)
N60 (UVW,XYZ)                   Activates the uvw parameters on PQD (the DAN has
                                renamed them in XYZ)

N70 X10u1Y20v0.3Z40w0.3
N80 (UPR,XYZ,PQD,0,20,20)       Activates a new UPR, the uvw are maintained on PQD
                                because these are still present


N90 X0u1Y5v0.50Z4w-0.5
```

MSA (UOV) - Defining a Machining Stock Allowance

The MSA command defines the value of the machining stock allowance along the profile. It is used in roughing and pre-finishing cycles. MSA can be programmed in a block, assigned in an MDI or entered via softkey.

Syntax

MSA = *value*

---

where:

value	It can be a decimal number or an E parameter. It is programmed in the same measuring unit (G70-G71) that is currently active in the program.

Characteristic:

The system uses the programmed MSA to calculate the offset value to be applied to the profile when cutter diameter compensation (G41 or G42) is active.

The offset value is the sum of the tool radius and the MSA.

Example:

```
MSA     = 0.5           Assigns a 0.5 machining allowance
E30     = 1.5
MSA     = E30           Assigns a 1.5 machining allowance
```

AUTOMATIC CONTOUR MILLING

The automatic contour milling commands (GTP and CCP) are designed for machining of profiles with radius compensation (G41/G42), generally defined using graphic tools.

The graphic tools (e.g. the Graphic Editor option) translate the drawing of the part to be machined into a technological program written in elementary ISO language.

One of the main characteristics of this feature is that the profiles for machining are seen as *subprograms*.

These profiles must be defined on a plane (e.g. XY) and can only contain the ***G*** operators for movement type (G1, G2, G3).

The contour milling commands will do the following:

- Automatically determine an approach point off the profile;
- Perform machining as described in the subprogram;
- Profile rotation, taking the first point of the profile as the rotation origin;
- Perform machining, starting from the last element of the subprogram through to the first (reverse machining).

In all cases, approach with compensation is scheduled on the first point of the profile (first block of program with abscissa/ordinate movements).

Limits to use of automatic contour miling

The limits regarding use of automatic contour milling are as follows:

- The profile must be described in full in a subroutine, indicated previously by *profile name*.
- The profile shall be considered closed if the first point described in the part program subroutine coincides with the last.
- Radius correction (*offset_mode*) will be activated automatically on the first block of the subroutine and de-activated on the last block.
- The blocks of the subroutine must be exclusively ISO type, programming of the following is allowed:
  - Axis names [dimensions]
  - *G* movement functions (*G1, G2, G3* with possibly their operands)
  - Three-letter commands in general shall NOT be taken into consideration (during the profile analysis stage), especially any commands that could modify the description of the profile (origins, mirror, scale factor).

NOTE:
Failure to respect these indications may lead to incorrect interpretation of the profile described in the subroutine.

GTP - Get Point

Determines the approach point off the profile.

Syntax

(**GTP,** profile_name, offset_mode, *E-par*, approach_type, angle)

---

where:

profile_name	An ASCII string representing the name of the part program that the profile is described in.
offset_mode	Possible values are 0, 1, 2. Is the tool diameter offset mode to be used: 0 = no offset 1 = offset with tool on left of profile 2 = offset with tool on right of profile
E-par	The variable in which the coordinates of the approach point calculated by the system will be loaded: index → abscissa index+1 → ordinate
approach_type	Possible values are 0, 1, 2, 3. Identifies the type of external approach to the profile; a different calculated point corresponds to each code (see next page).
angle	Angle of rotation: Represents the value of the rotation angle with which the profile will be executed.

NOTES:

- The distance between the external approach point to the profile and the machining starting point is determed on the basis of the value in system variable MSA.
- Activation is possible of a feature that inverts direction of motion of the profile using system variable TPO (TPO=4 profile inversion)
  For further details of the TPO coding values see page 4-17.

WARNING

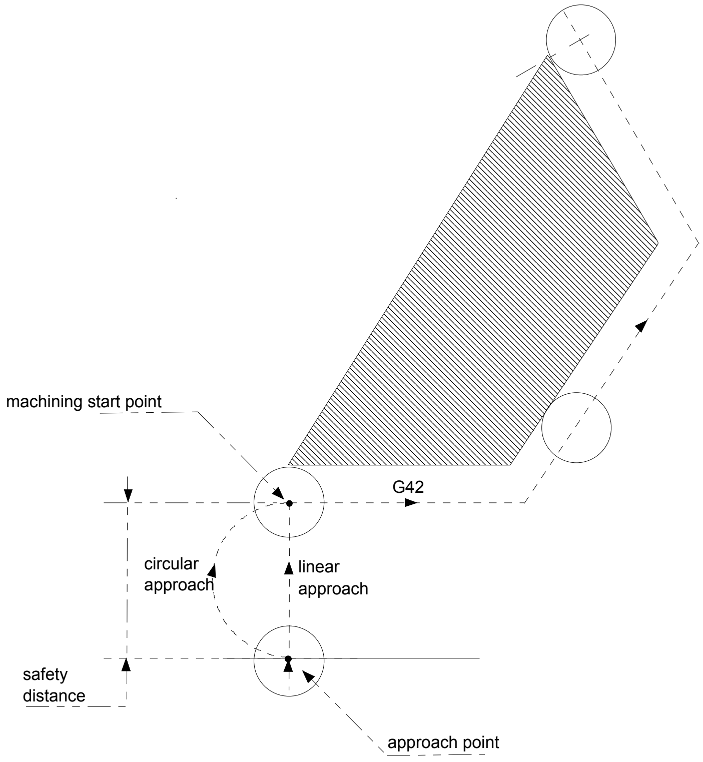
If you program offset_mode=0 , the approach_type is forced to “1” (approach tangential to profile) without any further indications from the system.

Determining the approach point

approach_type=0

Positioning occurs perpendicular to the first element of the profile.

The approach will be linear or circular, depending on value of the variable TPO.

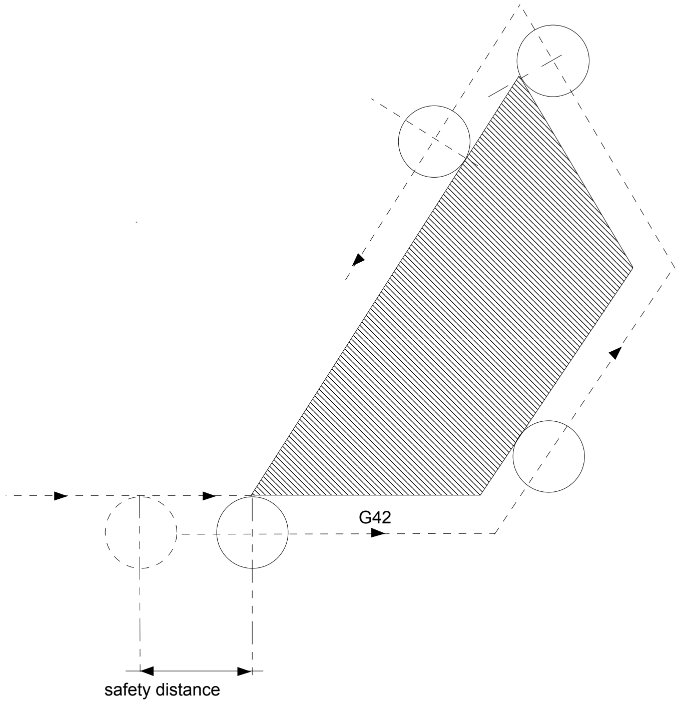


Perpendicular approach to profile

approach_type=1

Positioning occurs tangential to the first element of the profile.

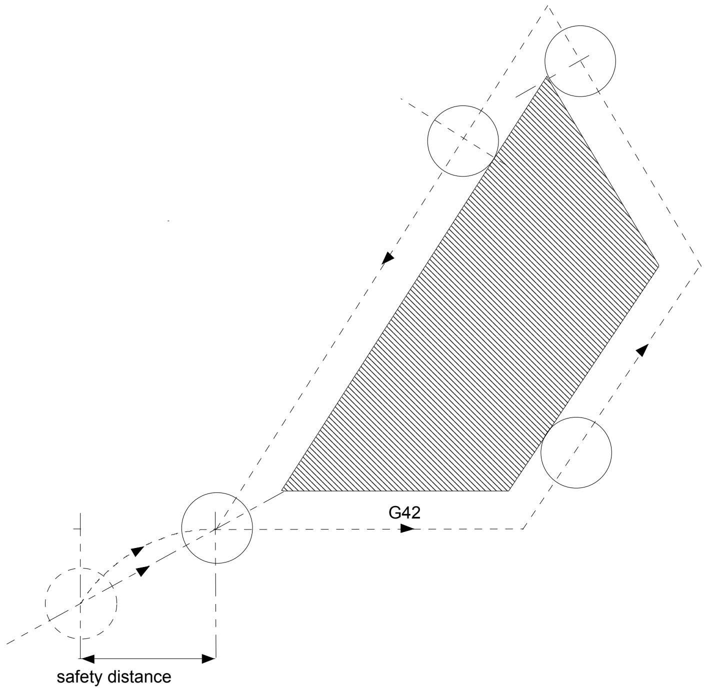
The approach is always linear.



Tangential approach to the profile

approach_type=2

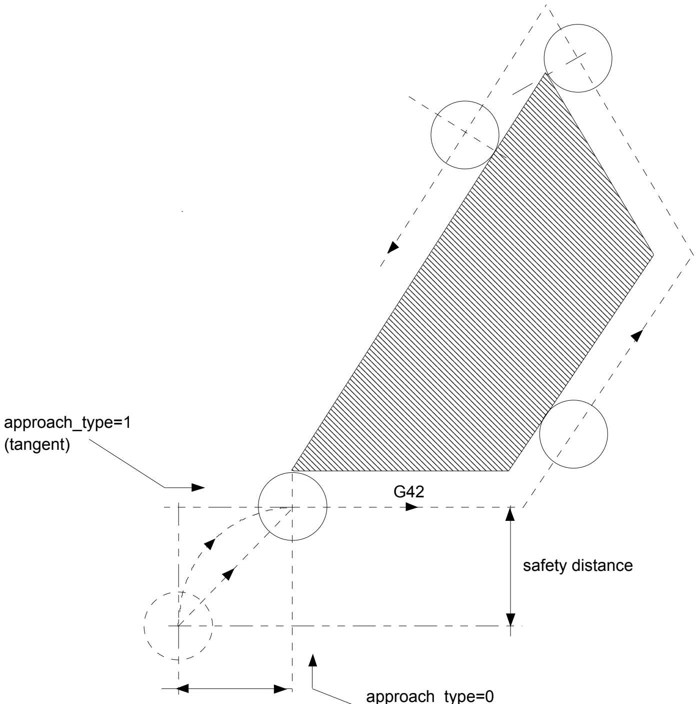
Positioning occurs on the intersection between the first and last points of the profile; the approach will be linear or circular, depending on value of the variable TPO.



Approach on intersection of first and last point of the profile.

approach_type=3

Positioning ocurs on the bisector of the angle formed by the movements generated with *approach_type=0 and approach_type=1*; the approach will be linear or circular, depending on the value of the variable TPO.



Approach on bisector

CCP - Cutter Compensation Profile

Performs contour milling of a profile, with tool diameter compensation.

Syntax

(CCP, profile_name, offset_mode, exit_type, angle)

---

where:

profile_name	An ASCII string representing the name of the part program the profile is described in.
offset_mode	Possible values 0, 1, 2. Represents the offset mode with which machining is performed.
exit_type	Possible values 0, 1, 2, 3. Identifies the type of exit from the profile; a different calculated point corresponds to each code.
angle	Angle of rotation. Determines the value of the rotation angle the profile will be executed with.

Characteristics:

The point of exit off the profile is determined by considering a value, seen as the safety distance.

This value, associated with the exit point type, determines position on the plane of the exit point taken from system variable MSA.

NOTE:
Activation is possible of a feature that inverts direction of motion of the profile using system variable TPO (TPO=4 profile inversion active).

For further details of the TPO coding values see page 4-16.

END OF CHAPTER

Chapter 5

PROGRAMMING THE SPINDLE

WARNING

The functions described in this chapter must be managed by the machine logic.

SPINDLE FUNCTIONS

With G96/G97 the S address programs the spindle speed in rpm or as constant surface feedrate. In addition, several M functions affect the spindle on/off state, range, etc.

G96 G97 - CSS and RPM Programming

The following G codes control spindle speed programming.

G96	Speed programming in constant surface speed
G97	Speed programming in rpm

Syntax

G97 [*G-codes*] [*operands*]

G96 [*G-codes*] [*operands*]

---

where:

G-codes	Other G codes that are compatible with G96 and G97 (see Table "Compatible G codes" in Chapter 1).
operands	Any operand or code that can be used in a G function block.

Characteristics:

G97 activates S programming in rpm. G97 is the default mode of the control and is modal with G96.

G96 activates constant surface speed (CSS) programming for the S word in feet per minute (G70), or metres per minute (G71). G96 forces the spindle speed to be controlled by the position of the diameter axis, so that it remains constant on the machining surface

When G96 is activated the position of the diameter axis is assumed to be the radius for which constant surface speed (S) is programmed. G96 is modal with G97.

Changing the S Word during G96. Cancelling CSS mode

If you want to change the S word value while G96 is active, the control must be in G00 or G29 mode. To cancel G96 CSS, the control must be in G00 mode and a block with G97 and an S word that defining the spindle rpm must be programmed.

Example:
The following example illustrates CSS programming.
Assumed power-up G codes: G00, G27, G70, G95, G97
Assumed gear range: 1 (M41) = 800 rpm max.

```
N1  G90 G00
N3  U5                ;Presets diameter (U) axis
N4  SSL=700           ;700 rpm limit
N5  G96 S400 M3       ;CSS at 400 ft /min
N6  G1 U0 Z -2 F10    ;Sets contouring mode and U in feed 10 ipm
N7  U5 Z0
N8  G00               ;Prepares to change S
N9  G01 S300 U0 Z-2   ;New surface feed = 300ft/min
N10 G00               ;Prepares to cancel G96
N11 G97 S100          ;Set G97, S=100 rpm
N12 G01 Z0            ;Contouring to G0
N13 U5
N14 G00               ;Rapid mode, contouring off
N15 M05               ;Stop spindle
N16 M02               ;Program end
```

IMPORTANT

S operators cannot be programmed when the process is on hold.

SSL - Spindle Speed Limit

The SSL command is used with G96 to set the maximum rpm that the spindle is allowed to run during CSS.

Syntax

SSL=*value*

---

where:

value	Is a value that can be programmed directly with a decimal number or indirectly with an E parameter.

Example:

SSL = 2000	Assign a spindle speed limit of 2000 rpm
E32 = 1500	
SSL = E32	Assign a spindle speed limit of 1500 rpm

Characteristics:

As the diameter axis approaches the spindle centreline, the spindle approaches maximum speed to maintain the programmed S word value. The SSL command limits the spindle to some value below the maximum rpm. If you program the value of SSL above the maximum rpm for the current gear range, the current gear range limit will be the maximum allowable rpm.

The SSL command must be programmed before the number of rounds **S**.

IMPORTANT Make sure you enter this value in the part program prior to entering G96 blocks.

M19 - Oriented Spindle Stop

Typically, the M19 function programs a spindle stop with a predetermined angular orientation. This feature is convenient in back spot-facing operations, because it allows to position the spindle, move the X or Y axis (depending on blade orientation), enter the hole, position the spindle again and start machining.

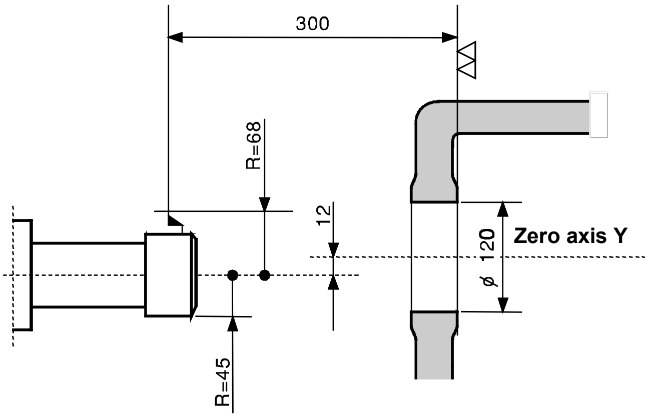
M19 can also be used for increased accuracy in boring operations, to avoid scoring the bored surface during the return move of Z axis. In this case, you would finish the hole, orient the spindle, move the X or Y axis according to the blade orientation, and perform the Z axis return.

M19 is deleted by M03, M04, M13, M14. When the control reads M19 in a block that also contains movement information, M19 precedes the movement.

IMPORTANT

The M functions that are implemented in your application may be different than those described here. Consult your machine documentation for more information on application-specific M functions.

Example:



Program:

```
N32 (DIS,"BACK SPOT- FACING BAR
D=136")
N33 S115 F20 T7.7 M6
N34 G X250 Y-12 M19
N35 Z-306
N36 Y M3
N37 DWT=2
N38 G1 G29 G4 Z-300
N39 G Z-302 M5
N40 Y-12 M19
N41 Z
```

In block N34 the X and Y axes are positioned and spindle orientation is controlled so that the tool can go through the hole. In the next block, the Z axis is positioned to start spot-facing.

In block N36 the tool axis is moved to coincide with the spot-facing axis.
Spot-facing is performed in block N38. In subsequent blocks, the tool is oriented, withdrawn from the workpiece and then positioned along the Y axis, so that it can go through the hole.

GTS – Spindle sharing

This three-letter code enables each process to release its spindle, change its status, or acquire a new axis in a new mode. For the various spindle conditions, see the AMP manual.

Syntax

(GTS)
(GTS, *spindle_id*, *mode*)

---

where:

spindle_id	Logic identifier of the spindle axis concerned.
mode	Mode requested, X = exclusive, S = shared.

(GTS)

This is used to release a spindle. With this syntax the process tells the system it does not want to use the spindle any longer and the spindle is made available for use by other processes. Axis status is changed to RELEASED only if the release is performed by the last owner process; if an axis is still being used by other processes, its status continues to be SHARED.

(GTS, *spindle_id*, S)

This is used to share a spindle with other processes with id *spindle_id* which must not be used by another process on an exclusive basis. Possible cases include:

1) A process does not own any spindle and wants to acquire a new spindle in shared mode.
2) A process owns the axis on an exclusive basis and wants to make it available for sharing.
3) A process wants to release the current spindle and acquire a new one.

(GTS, spindle_id, X)

This is used to use a spindle with id *spindle_id* on an exclusive basis. The axis will not be shared with other processes. Possible cases include:

1) A process does not own any spindle and wants to acquire one for exclusive use.
2) A process is the only owner of a shared spindle and wants exclusive control of it.
3) A process wants to release the current spindle and acquire a new one.

Non envisaged cases generate a process error.

Example:

(GTS, 4, S)

Characteristics

The second field (spindle id) can be a variable; the third field (mode) can only assume the values described. Moreover, a call to the entire part program forces the block-calculation synchronism.

END OF CHAPTER

Chapter 6

MISCELLANEOUS FUNCTIONS

Standard M functions

In this chapter we briefly describe the miscellaneous functions (hereinafter M functions) that are considered standard for most applications. These functions are listed in the table below.
Since the machine integrator has programmed the interface between the control and your application, the M functions that are implemented in your machine may be different than those described here. Consult your machine documentation for more information on application-specific M functions.

M FUNCTION	ACTIVE PRE	ACTIVE POST	CANCELLED BY	MEANING
M0		x	Cycle Start	Program Stop
M1		x	Cycle Start	Optional Program Stop
M2		x		End of Program
M3	x		M4-M5-M14-M19	Spindle CW
M4	x		M3-M5-M13-M19	Spindle CCW
M5		x	M3-M4-M13-M14	Spindle Stop
M6		x		Tool Change
M7	x		M9	Auxiliary Coolant On
M8	x		M9	Main Coolant On
M9		x	M7-M8	Coolant Off
M10	x		M11	Axes Lock
M11	x			Axes Unlock
M12	x		M11	Rotary Axes Lock
M13	x		M4-M5-M14-M19	Spindle CW and Coolant On
M14	x		M3-M5-M13-M19	Spindle CCW and Coolant On
M19	x		M3-M4-M5-M13-M14	Spindle Stop and Angular Orientation
M30		x		End of Program and Reset to 1st Block
M40	x			Deactivate Spindle Range
M41	x		M42-M43-M44-M40	
M42	x		M41-M43-M44-M40	Spindle Range
M43	x		M41-M42-M44-M40	1-2-3-4
M44	x		M41-M42-M43-M40	
M45	x		M41-M42-M43-M44	Automatic Range Change
M60		x		Part Change

In the AMP environment you may declare that certain M functions can be disabled by a control reset.

IMPORTANT

The block that includes an expedite M code must also program an axis move.

Depending on how the M has been configured, the move may be point-to-point (G29) or continuous (G27-G28).

M CODE	DESCRIPTION
M0	**Program Stop**: M0 stops program execution, spindle rotation and coolant flow after the control has performed all the operations of the block in which it appears. The control retains all current status information after executing an M0.
M1	**Optional Program Stop**: If the command is enabled using the appropriate softkey, M1 operates like the M0 code.
M2	**End of Program**: M2 defines the end of a program.
M3	**Spindle Rotation Clockwise**: M3 defines spindle clockwise rotation. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled before any other axis move.
M4	**Spindle Rotation Counter clockwise**: M4 defines spindle clockwise rotation. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled before any other axis move.
M5	**Spindle Stop**: M5 stops spindle rotation. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled after any axes moves.
M6	**Tool Change**: M6 temporarily stops the system from reading the program. It activates the offsets selected by the T function. In a block it is enabled before any other axis move. If it also stops spindle rotation and coolant flow, then M6 does not disable M3, M4, M7, M8, M13, M14. These functions become active again after M6 is completed.
M7	**Auxiliary Coolant On**: M7 turns the auxiliary coolant on. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled before any other axis move
M8	**Main Coolant On**: M8 turns the main coolant on. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled before any other axis move.
M9	**Coolant Off**: M9 turns all coolant systems off. It is enabled by the control either when it is entered from the keyboard or when it is read from a program. In a block it is enabled before any other axis move.

M functions (cont'd)

M CODE	DESCRIPTION
M10	**Linear and Rotary Axes Lock**: M10 locks the axes that are not involved in the current machining process.
M11	**Deactivates M10 and M12**
M12	**Rotary Axes Lock**: M12 locks only the rotary axes that are not involved in the current machining process.
M13	**Spindle Rotation Clockwise and Coolant On**
M14	**Spindle Rotation Counter clockwise and Coolant on**
M19	**Oriented Spindle Stop**: In a block, M19 spindle is oriented before any motion in the block. M19 deactivates M3, M4, M13, M14.
M30	**Automatic Reset at End of Program:** M30 deletes all information from the control dynamic buffer. Absolute origin 0 is automatically enabled and the selected program is set for restart. M30 does not deactivate the offset of the tool in the spindle.
M40	**Spindle Range Reset**
M41	**Spindle Range 1**
M42	**Spindle Range 2**
M43	**Spindle Range 3**
M44	**Spindle Range 4**
M45	**Automatic Spindle Range Change**
M60	**Workpiece Change**

END OF CHAPTER

PARAMETRIC PROGRAMMING

Parametric programming requires the programmer to use system and local variables.

System variables are stored in the dual port memory of the system. They are seen by all the active processes and are retained after the system is switched off.

Local variables are stored in a memory area local to the system and are seen only by the process they refer to. Their value is lost when the system is switched off. At power up they are re-initialised with the value defined in AMP.

The following table summarises the variables available with the system.

VARIABLE	TYPE	FUNCTION
E	local	Variables
!*name*	local	User variables
SN	system	System number
SC	system	System character
TIM	system	System timer (read only)
@*name*	system	PLUS variables

Except for System Characters, all system and local variables can be used in mathematical or trigonometric operations in place of the geometrical and technological data of the machining cycle.

A mathematical operation is formed by arithmetic operators, functions and operands (variables or numerical constants). The following are arithmetic operators:

- + addition
- - subtraction
- * multiplication
- / division

TRIGONOMETRIC FUNCTIONS

Trigonometric functions used by the system are listed in the table below.

FUNCTION	DESCRIPTION
SIN (A)	sine of A
COS (A)	cosine of A
TAN (A)	tangent of A
ARS (A)	arc sine of A
ARC (A)	arc cosine of A
ART (A)	arc tangent of A
SQR (A)	square root of A
ABS (A)	absolute value of A
INT (A)	integer portion of A
NEG (A)	negation of A
MOD (A,B)	remainder of A to B ratio

The arguments of a function (A,B) can be variables or numerical constants. When the control solves a mathematical equation, it considers the priority of brackets and signs. The result is converted into the format of the variable written to the left of the equal sign.

IMPORTANT

Arguments of trigonometric functions (SIN, COS, TAN) must be expressed in degrees. The result of inverse trigonometric functions (ARS, ARC, ART) is also expressed in degrees.

BOOLEAN FUNCTIONS

The boolean functions available with the system are listed in the table below.

FUNCTION	DESCRIPTION
AND(A,B)	Executes AND at the bit between two numbers in the -32768 +32767 range*
OR(A,B)	Executes OR at the bit between two numbers in the -32768 +32767 range*
NOT(A)	Complement to 1 of a number in the -32768 +32767 range*

IMPORTANT

* The parameters of boolean functions must be integers in the -32768 +32767 range (short format with sign). They may be long real E parameters as long as they remain in this range.

GTL FUNCTIONS

GTL functions available with the system are listed in the table below.

FUNCTION	DESCRIPTION
FEL(A,B)	Calculates the element having a B(1,2,3) index from the straight line whose index is A(1=sine of the angle, 2=cosine of the angle, 3=distance from origin to the straight line). **Example:** E30=FEL(5,1) assigns to E30 the sine of the angle formed by the l5 straight line and the abscissa.
FEP(A,B)	Calculates the element having a B(1,2) index from the point whose index is A(1=point abscissa, 2=point ordinate). **Example:** E34=FEP(4,2) assigns to E34 the ordinate of the p4 point.
FEC(A)	Calculates the element having a B(1,2,3) index from the circle whose index is A(1=center abscissa, 2=center ordinate, 3=circle radius). **Example:** E42=FEC(8,3) assigns to E42 the radius of the c8 circle.

LOCAL VARIABLES

E Parameters

The maximum number of E parameters must be defined during system configuration. In theory, there can be up to 8000 E parameters.

E parameters are of the Long Real type, which allows 15 digits in total, 12 maximum before the decimal point and 9 decimals. The system accepts several statements per block, the only restriction being block length. When in block-by-block mode, multiple statements will be executed as if they were in a single block. Two levels of parametric indexes are allowed. For example: E(E(E..)).

E parameters receive values in special assignment blocks. The format of an assignment block is:

E*n* **=** *expression*

---

where:

n	Is the identification number of the E parameter.
expression	Can be a numerical value, a character or a mathematical equation whose result is stored in the E parameter identified by *n*.

Examples:
The following are assignment blocks for parameter calculation.

E37=(E31*SIN(E30)+123.4567)/SQR(16)	Solves the mathematical equation and assigns the result to parameter E37.
E39=-0.00000124+5	Calculates the expression and assigns the result to parameter E39.
E40=TAN(35)	Finds the tangent of 35° and assigns the result to parameter E40.
E31=NEG(E31)	Changes the sign of parameter E31.
E7=81	Assigns the value 81 to parameter E7.
E25=E25+30	Adds 30 to the current value of E25 and assigns the result to E25.
E29=1,2,3,4,5	Assigns the value 1 to parameter E29, the value 2 to parameter E30, the value 3 to parameter E31, the value 4 to parameter E32 and the value 5 to parameter E33.

E parameters can be used inside programs and subroutines. To display the current value of an parameter, use the DIS command. For instance,

Example:

(DIS,E39)	displays the current value of E39.

Example:
This example shows how to assign an ASCII character:

SCO="P"	Assigns the P character to the SC0 string variable
E1=SC0	
(DIS,E1)	Displays 80 (ASCII code of P)

Examples of motion blocks or commands with parameters.

```
XE1
X-E1
X(E1)
X(-E1)
X(E8-14*SQR(E14))
X(-(E8-14*SQR(E14)))
X(E(E(E3)))

FE1
SE2
TE1.E2
```

! - User Variables

User defined variables can be of two types:

- Long Real
- Character

User variables must be defined in the AMP configuration. A user variable name can be up to 8 characters long. The first character must be !. The extension of a user variable may be .LR or .CH.

With user variables of the character type apply this rule:

!name_var[(*index*)].[*number of characters*]CH = Parameter

where:

index	Number indicating the starting character in the variable character array. If index is not specified, it is taken to be zero. If specified, it must be programmed between round brackets.
number of characters	Specifies how many characters after the index must be read/written. The default value is 1. The sum of index+number of characters must not be greater than the maximum number of characters configured for the specified variable.
parameter	Can be: – a string constant enclosed between apexes or quotes; – a string variable not longer than *length* – a numerical constant in the 0 - 255 range – a numerical variable in the 0 - 255 range.

Example:
!ABC(1) = 125
G0 X(!ABC(1))

125 is assigned to the !ABC(1) user variable, then this variable is used as argument of the X address in a G0 block.

!CHAR(2).8CH="ABC"

This instruction writes "ABC" in the first three characters of the !CHAR user variable, starting from the second character. The remaining 5 characters (8-3) will be automatically set to zero; to prevent this, program !CHAR(2).3CH="ABC"
!CHAR(1).CH="A" or!CHAR(1).CH=65

Long Real type user variables have the following format:

!name_var[(*index*)]= Expression

---

where:

index	Number identifying the variable. If *index* is not specified, it is taken to be zero. If specified, it must be programmed between round brackets.
expression	Can be a numerical value or a mathematical expression, whose result is stored in the user variable identified by the index.

Example:

```
!ABC(1) = 125
G0 X(!ABC(1))
```

125 is the value assigned to the !ABC(1) user variable; this variable is then used as an argument of address X in a G0 code.

NOTES:
The index of the variable can be a number or an E parameter.
In motion blocks, the user variable must always be written in brackets.

Examples of motion or command blocks with Long Real user variables:

```
X(!USER1(2))
X(!USER1(2)∗10)
F(!USER1(1))
S(!USER1(1))
T(!USER1(1).(!USER1(2))
```

NOTES:
The index of the variable may be a number or an E parameter.

SYSTEM VARIABLES

There are four types of system variables that can be used in part programs:

- System Number
- System Character
- System timers
- Plus variables

These variables can be used to read or write values or strings for assignment operations within part programs.

SN - System Number

System Number variables are of the Long Real type, which allows 15 signed digits with 12 integer digits maximum. Up to 25 System Numbers can be defined in the 200 byte area that is available in the dual port memory of the system. These variables do not allow index levels.

The format of a System Number variable is as follows:

SN*n***=** *expression*

where:

n	Is the identification number of the System Number variable. The n parameter can be a number or an E parameter.
expression	Can be a numerical value, an equation, or a character whose result is stored in the System Number identified by *n*.

NOTE:
You can assign one System Number variable to each defined System Number.

Examples:

SN20 = 326.957	the decimal value 326.957 is assigned to the SN20 variable.
SN20 = (SN9*SIN(30) + 12.5)/SQR(81)	the result of the mathematical expression is assigned to the SN20 variable.

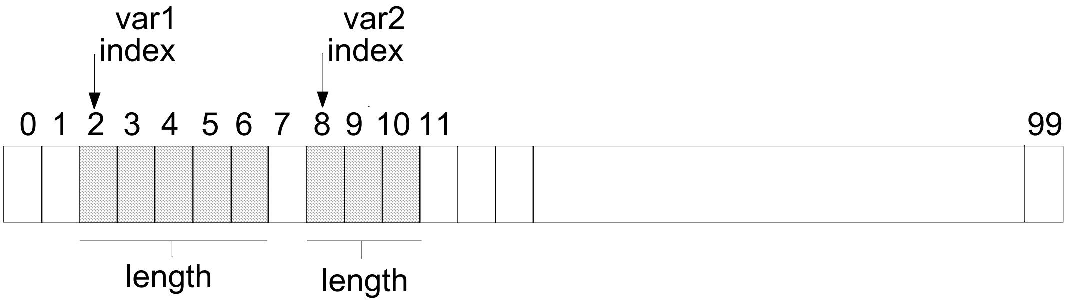
Examples of motion or command blocks with SN variables:

```
X(SN0)
X(SN0·∗SN1)
F(SN1)
S(SN2)
T(SN1).(SN3)
```

SC - System Character

System Character variables are of the character type and are stored in the dual port memory of the system, where a 100 byte array is reserved for them. This means that all the defined System Characters cannot occupy more than 100 bytes.

Each System Character is identified by an index that specifies the start address inside the array and by a length that specifies how many bytes the variables occupies starting from that address.



The format of a System Character variable is as follows:

SC*index***.***length* **=** *parameter*

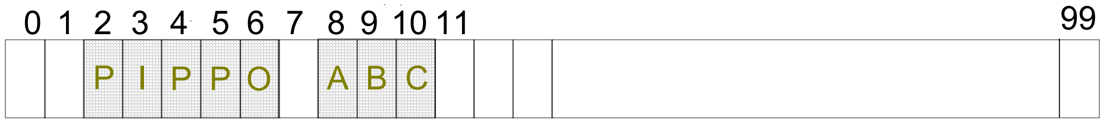
where:

index	Is the index that specifies the start position of the variable in the array. It may be in the 1 to 100 range. The index of the variable can be a number or an E parameter.
length	Is the length of the variable expressed in number of characters (bytes). The maximum allowed length for a single variable is 80 characters. The length can be a number or an E parameter.
parameter	A parameter may be: - a string constant enclosed between apexes or quotes; - a string variable not longer than *length* - a numerical constant in the 0 - 255 range - a numerical variable in the 0 - 255 range.

NOTES:

- For each variable, the i*ndex+length* sum must not exceed 100.
- One System Character variable can be assigned to another System Character variable.
- The SC variable may also be assigned the numeric variables (decimal) corresponding to ASCII characters. In this way it is also possible to assign those characters that are not displayed. For example, 10 (LF) and 13 (CR).
- Numerical values are programmed without double quotes (" ").

Example:
SC3.5="PIPPO"
SC9.3="ABC" or SC9.3=65,66,67



The string "PIPPO" is written starting from byte 3 of the array and occupies 5 bytes.

The string "ABC" is written starting from byte 9 of the array and occupies 3 bytes.

IMPORTANT When defining the index and the length, care must be taken not to overlap two variables.

Example:

SC0.1=80	assigns 80 to the string variable
(DIS,SCO.1)	displays P (ASCII code 80)
E1=80	assigns 80 to E1
SC0.1E1	
(DIS,SC0.1)	displays P (ASCII code 80)

TIM - System Timer

The TIM instruction defines a variable used by the programmer to read the time shown by the control timer. Its value is expressed in seconds.

A TIM instruction can be read, displayed or stored in a support variable. Computation of this value starts as the control is switched on.

Example:

```
(DIS, TIM)
E10=TIM
```

NOTE:
TIM contents cannot be modified from part program.

@ - PLUS Variables

From part-program it is possible to read or write the PLUS variables managed by the machine logic.

Syntax

@name

where:

name Is the name of the variable declared in the configuration (AMP).

There are three types of PLUS variables:

- Short
- Boolean
- Double

There are 256 Short variables, but the part program can only access 128 of them. The names of the Short variables that can be accessed by a part program must be configured in AMP. They are made up of 16 bits and can contain values between -32768 and +32767.
You can also address a single bit of a Short variable with a Boolean variable. There are 256 Boolean variables, but a part program can only access 128 of them. The names of the Boolean that can be accessed by part programs must be configured in AMP.

There are 64 Double variables, all accessible by the part program. The names of the Double variables that can be accessed by a part program must be configured in AMP.
The total number of Boolean, Short and Double variables must be configured in AMP.

The values of all the PLUS variables configured in AMP are loaded when the system is started up. If the *value* field of a variable is empty, the variable is not initialised and retains the value it had when the system was switched off.
PLUS variables can be used in a part program as follows:

1. Assignment blocks and three-letter codes
   ```
   E10=@LOG1
   (GTO,END,@LOG3=1)
   ```
2. Motion or command blocks
   ```
   G0 X (@LOG2)
   X(@LOG2∗2)
   F(@LOG2)
   S(@LOG3)
   T(@LOG2)·(@LOG3)
   ```

WARNING

Because these variables are directly linked to the machine logic, the machine tool manufacturer must provide you with the list of the variables used in your application as well as with all the information you need in order to handle them correctly.

L Variables

L variables have a long real format, with 15 signed figures and a maximum of 12 integer figures. There are 400 L variables indexed from 0 to 399.

These variables conform with the user tables available in the table editor and the PLUS environment. They can be utilised both in the programming and the logic environment, either separately or for communications between both environments.

Examples:

```
L10 = 26.9570
L15 = (L10*SIN (30)+9)/SQR(81)
(GTO,END,L2=1)
G0XL15
X(L15)
X(L15∗L1)
FL1
SL1
TL1.L3
```

WARNING

Because these variables are directly linked to the machine logic, the machine tool manufacturer must provide you with the list of the variables used in your application as well as with all the information you need in order to handle them correctly.

Multiple Assignments

With the multiple assignment operators **{ }** it is possible to assign the contents of other variables to a certain number of variables.

Multiple assignment is accepted only for numeric variables.

Syntax

destination_variable = **{** *source_variable, number_variables* **}**

where:

destination_variable	Is the first destination variable.
source_variable	Is the first of the source variables.
number_variables	Is the number of variables to transfer. This can be an integer or a local or system variable.

Examples:
E0 = **{** SNO,4 **}**

is equivalent to the following four assignments:

E0 = SN0
E1 = SN1
E2 = SN2
E3 = SN3

E100 = 5
E50 = **{** LO,E100 **}**

is equivalent to the following five assignments:

E50 = L0
E51 = L1
E52 = L2
E53 = L3
E54 = L4

END OF CHAPTER

CANNED CYCLES

CANNED CYCLES G8N

Codes G81 through G89 define canned cycles that let you program multiple operations (drilling, tapping, boring, etc.) without repeating parameters and commands that are common to all the operations.

In the block programming the G81-G89 canned cycle it is not possible to also program axes moves. The cycle is stored, but not executed.

Canned cycle execution starts from the block that follows the definition of the G81-G89 block. To repeat a cycle a second time you must simply program the coordinates of thet points at which the cycle must re-start.

The spindle axes for the canned cycle can be assigned in the G81-G89 block.

For example, in the G81 R Y-20 block the Y axis is the spindle of the canned cycle.

G8n functions are modal. Before programming a new canned cycle you must first cancel the current one with G80. The G80 function must be programmed in the block that follows the last canned cycle.

You cannot program a G8n block if cutter diameter compensation (G41/G42) is active.

When a canned cycle such as G82, G83, G89 requires a dwell time you can:

- use the default dwell time defined in AMP
- program a block containing the variable DWT=time (expressed in seconds)

IMPORTANT Canned cycles can also be performed on virtual axes.

Canned Cycle Features

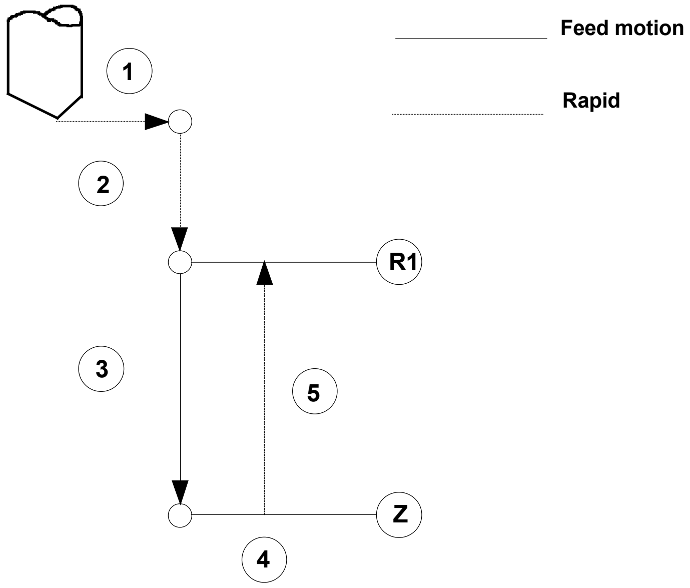
This table lists canned cycles available with 10 Series and their features.

CANNED CYCLE	INFEED	DWELL	ROTATION	RETURN
G81 drilling	working	no	normal	rapid
G82 spot-facing	working	yes	normal	rapid
G83 deep drilling with chip take out	intermittent working (down at machining rate spaced with rapid retracts	no	normal	rapid
G84 tapping	working spindle rotation start	no	rotation reversal	working to R1 rapid to R2 if present
G85 reaming or tapping by Tapmatic	working	no	normal	working to R1 rapid to R2 if present
G86 boring	working spindle rotation starts	no	stop	rapid
G89 boring with spot facing	working	yes	normal	working to R1 rapid to R2 if present
G80 deletes the canned cycle				

Canned Cycle Moves

When the system reads a canned cycle programmed in a block the axes execute the following sequence of moves:

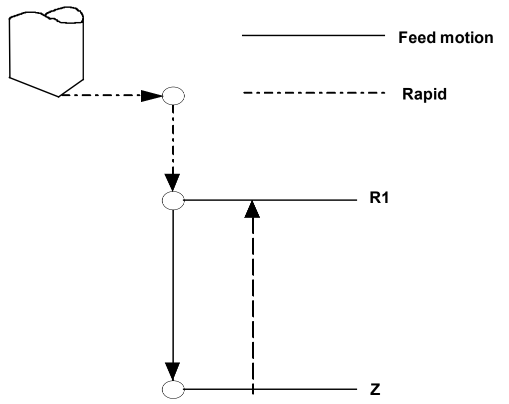
1. Rapid positioning to the centre line of the hole
2. Rapid approach to the work plane (R1 dimension)
3. Machining feedrate down to the programmed depth dimension (Z)
4. Cycle functions at the bottom of the hole
5. Rapid or machining feedrate return to the R dimension (R2 if the return dimension is different from the approach R1)



Examples of Canned Cycles

Two typical canned cycles are shown in this section.

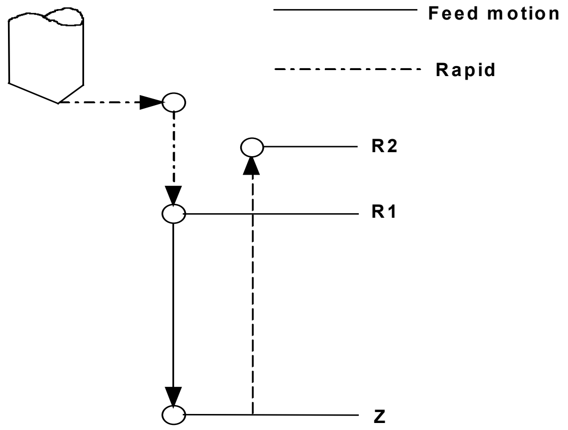
The following is a canned cycle in which the approach coordinate is equal to the return coordinate.



G81 R_1 . . Z. .

R_1 = approach coordinate
Z = hole depth

In this canned cycle the return position is different from the approach position.



G81 R_1. . R_2. . Z. .

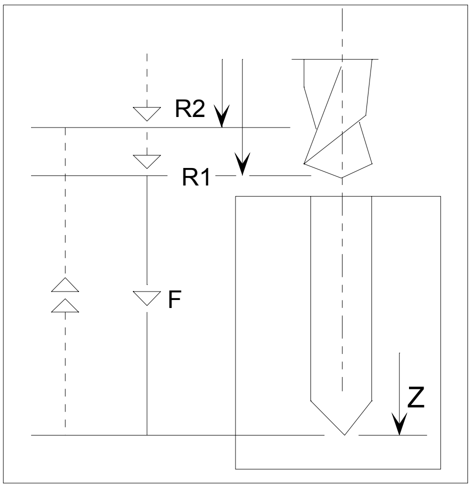
R_1 = approach coordinate
R_2 = return coordinate
Z = hole depth

G81 - Drilling Cycle

Syntax

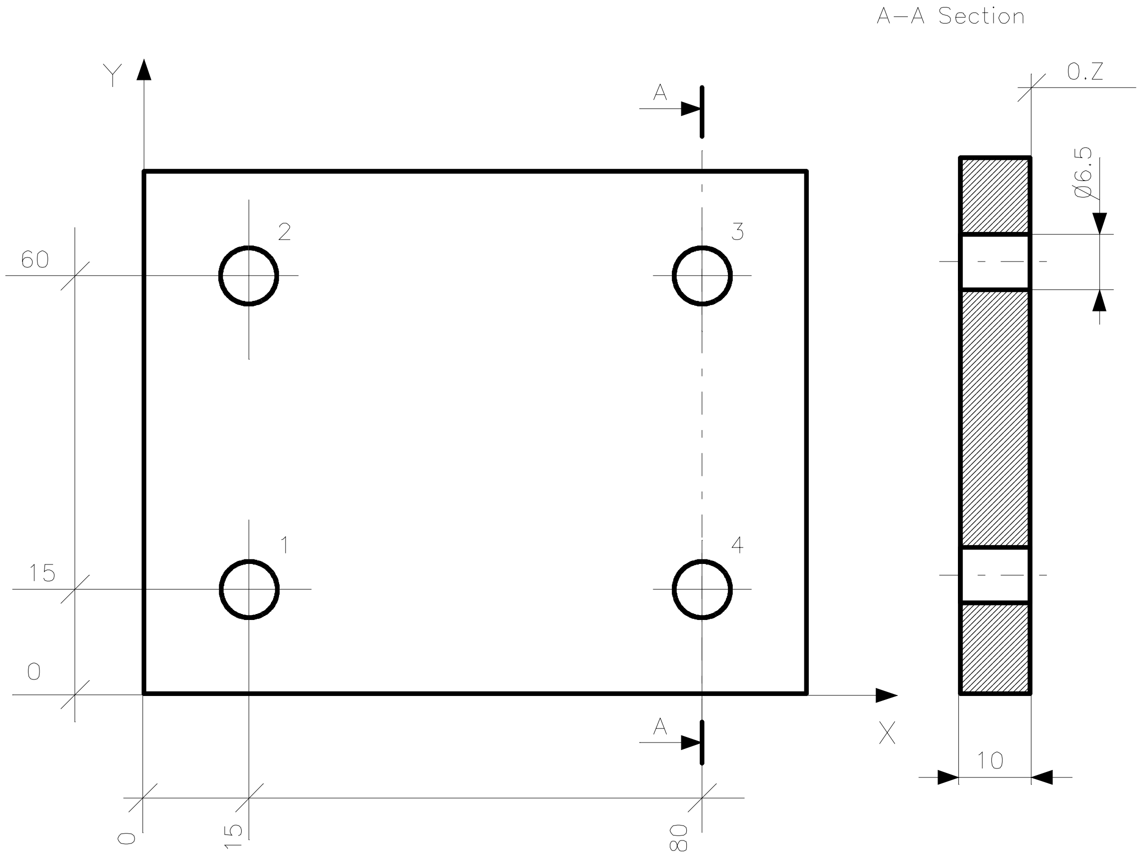
G81[*G-codes*] *R1.. [R2..] Z.. [F..]* [*auxiliary*]

where:



G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Approach dimension (mandatory). Defines the coordinates for rapid positioning on the machining plane when the cycle starts. The R address is followed by the rapid approach value. It can be programmed directly with a decimal number or indirectly with an E parameter.
R2	Defines the coordinates for return after machining (R planes). The R address is followed by the return value. If it is omitted, the control assumes by default the approach dimension (R). R2 can be programmed directly with a decimal number or indirectly with an E parameter.
Z	Defines the hole depth (typically Z). The Z address is followed by the depth value, which can be programmed directly with a decimal number or indirectly with an E parameter.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.
auxiliary	M, S or T programmable auxiliary functions. In a canned cycle block you can program up to four M functions, one S (spindle speed) and one T (tool selection).

Example:



NOTE:
0.Z means Z=0

Program:

```
    (UGS,X,0,110,Y,0,80)
    N31 (DIS,"TWIST DRILL D=6.5")
    N32 S1100 T3.03 M6
    N33 G81 R3 Z-15 F95 M3
1   N34 X15 Y15
2   N35 Y60
3   N36 X80
4   N37 Y15
    N38 G80 Z50 M5
    N39 M30
```

G82 - Spot Facing Cycle

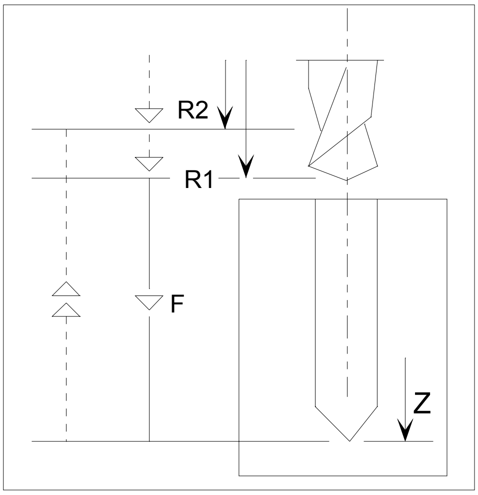
Syntax

G82[*G-codes*] *R1.. [R2..] Z.. [F..]* [*auxiliary*]

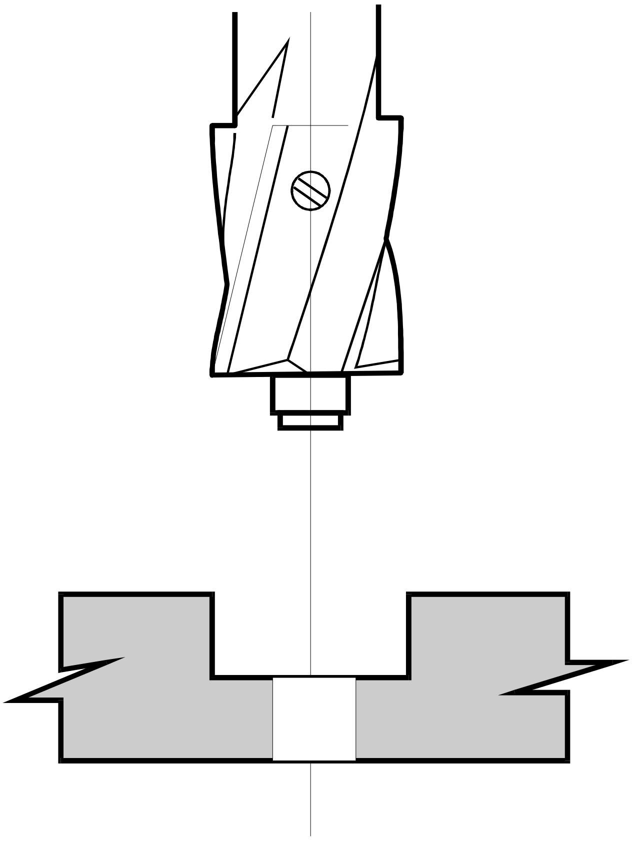
---

where:

G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the value of hole start (see G81).
R2	Defines the coordinates for return (see G81).
Z	Defines the value of hole end (see G81).
F	Defines the value of feedrate(see G81).
auxiliary	M, S or T programmable auxiliary functions.



Example:
The following figure shows a spot facing cycle.

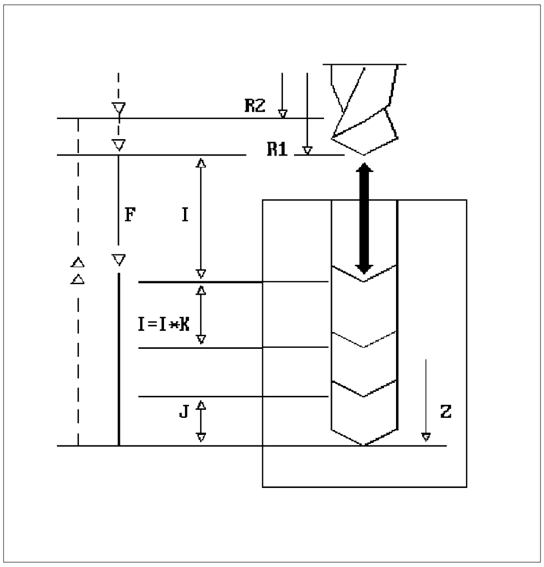


```
G82 R5 Z-15 F.. M..
X.. Y..
```

G83 - Deep Drilling Cycle

Syntax

G83 [*G-codes*] *R1.. [R2..] Z.. I.. [J..] [K..] [F..]* [*auxiliary*]



where:

G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the value of hole start (see G81).
R2	Defines the coordinates for return (see G81).
Z	Defines the value of hole end (see G81).
I	Defines the depth increment after each pull-out in for wood pecking cycles. The I address is followed by the depth increment value.
J	Defines the minimum depth increment after which the cycle applies a constant increment. The J address is followed by the minimum depth increment value.
K	Defines the reduction factor applied to *I* until the *J* value is reached. The K address is followed by a numerical value.
R1	Defines the value of hole start (see G81).
R2	Defines the coordinates for return (see G81).
Z	Defines the value of hole end (see G81).
F	Defines the value of feedrate (see G81).
auxiliary	M, S or T programmable auxiliary functions.

Characteristics:

With G83 different moves may be generated no matter whether **I, J** and **K** are programmed in the block.

If **I,J** and **K** are programmed axes moves are as follows:

1. Rapid approach to the hole centre line
2. Rapid approach to the **R** coordinates
3. Machining feedrate to the **R** + **I** coordinates
4. Rapid retract to the intial **R** coordinates each time a chip discharge occurs.
5. Calculation of the new **R** value with the formula:

   $R = R_{old} + I - X$ (where X is 1 by default, or, if the DRP variable is programmed, it assumes the value assigned to this variable)
6. Calculation of the new I value with the formulas:

   $I = I_{old} * K$ if $I * K > J$

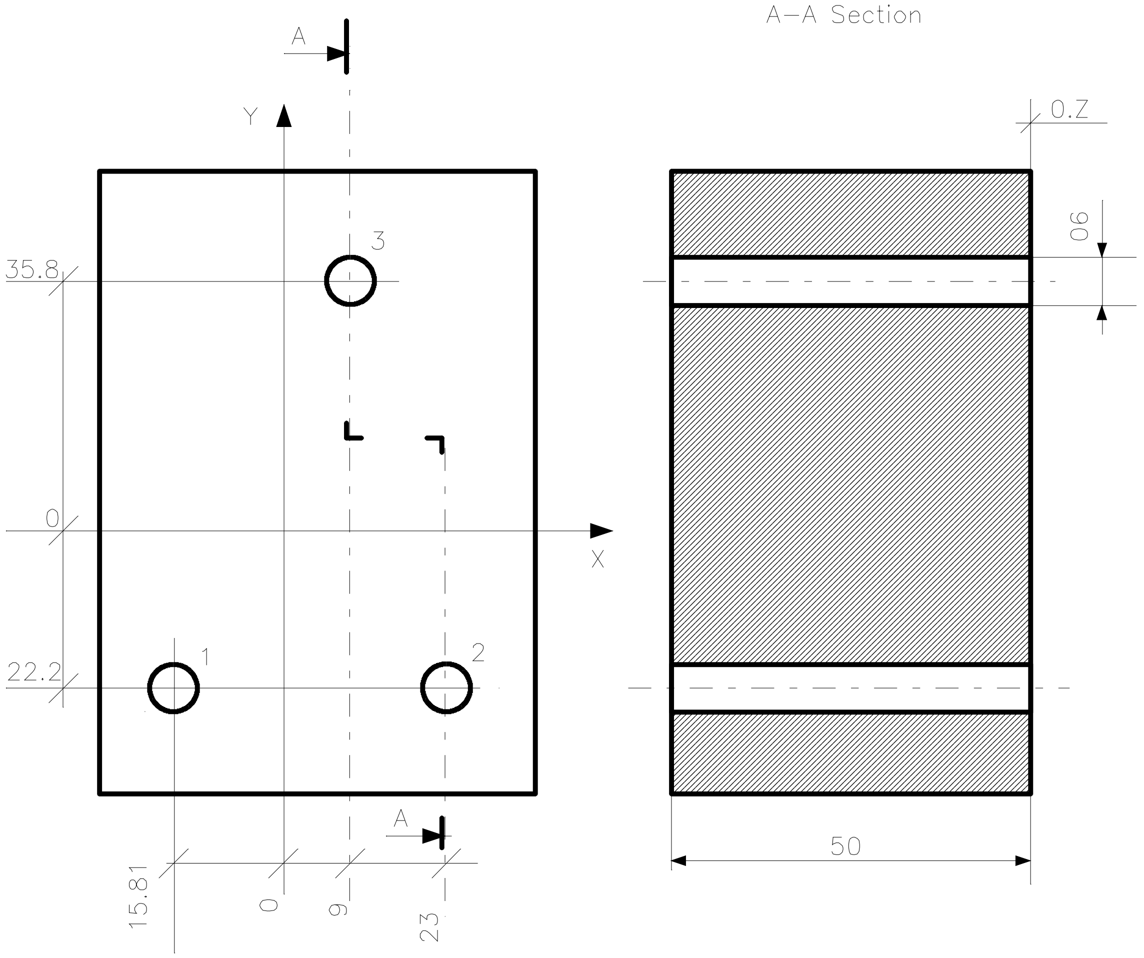
   $I = J$ if $I * K < J$
7. Repetition of points 2 through 6 until the Z dimension is reached.

Chip breaking cycle

If J and K parameters are not programmed, the following movements are generated:

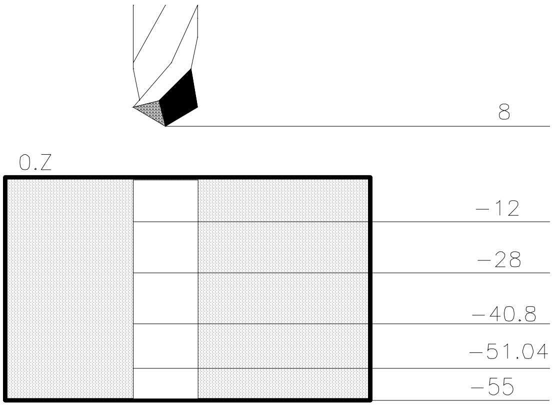
1. Rapid approach to the hole centre line
2. Rapid approach to the R coordinates
3. Machining rate to R + I
4. Spindle dwell for the time programmed in the DWT function, or for the characterized time if DWT is not programmed.
5. Repetition of points 3 and 4 until the Z dimension is reached.

Example:



NOTE:
0.Z means Z=0

Program:



```
    N65 (DIS,"TWIST DRILL D=6")
    N66 S930 F65 T6.6 M6
    N67 G83 R8 Z-55 I20 K.8 J6 M13
1   N68 X-15.81 Y-22.2
2   N69 X23
3   N70 X9 Y35.8
    N71 G80 Z50 M5
    N72 M30
```

DRP – G83 hole reworking distance

The DRP command defines in mm the hole reworking distance in G83 cycles (with IJK versors programmed).
This value can be initialised at 1 by the system. It can be assigned from the Keyboard or the Part Program.

Syntax:

DRP=value

---

where:

value	can be programmed directly or indirectly with an E parameter.

Example:

DRP=0.01	hole reworked at 0.01 mm
DRP=5	hole reworked at 5 mm

WARNING

RESET restores the value of 1.

G84 - Tapping Cycle with no Transducer

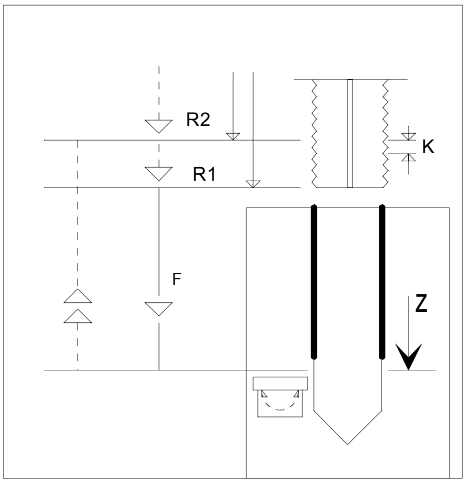
This G84 code operates when a transducer is not mounted on the spindle. The spindle must belong to a process in exclusive mode.

Syntax

G84 [*G-codes*] *R1.. [R2..] Z.. [F..]*

where:

G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the rapid approach coordinates and the the fixed return to work.
R2	Defines the rapid return coordinates.
Z	Defines the hole depth (typically Z). The Z address is followed by the depth value, which can be programmed directly with a decimal number or indirectly with an E parameter.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.



Characteristics:

The tool that is approaching the workpiece at rapid must stop at five times the tap pitch (for depths less than 3.O) or seven times the tap pitch (for depths greater than 3.O) from the workpiece.

The tapping feedrate must be calculated with the following formula:

$$F = S * p * 0.9$$

where:

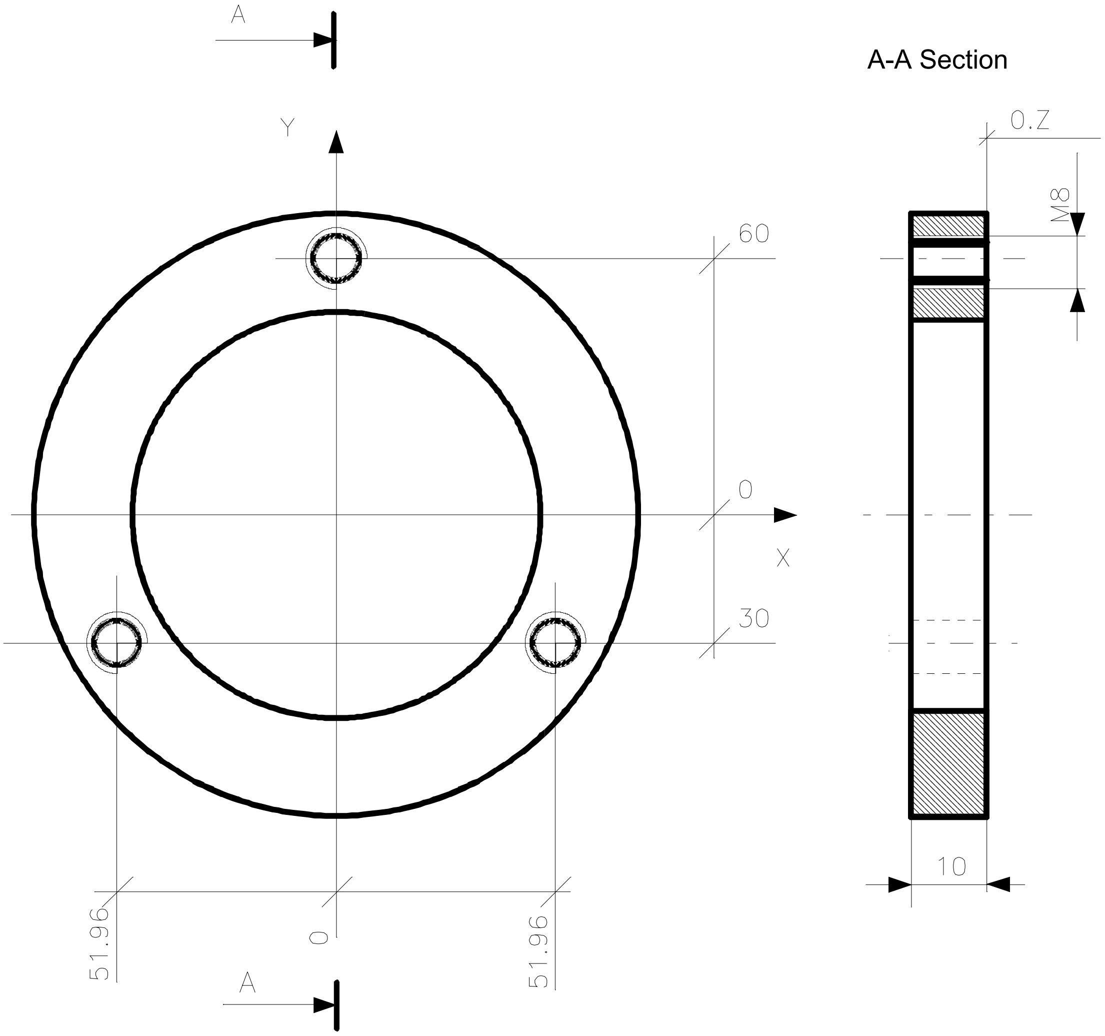
S	spindle rotation speed
p	the tap pitch
0.9	the feedrate decrease factor to keep the tool holder spring compensator stretched

The final Z must be decreased by 10% of the actual tap working travel.

The final Z must be long enough for the axis to reach the programmed feedrate and stop with controlled deceleration. It must be calculated according to the time it takes spindle rotation to stop. If the final Z is not long enough, the control displays an error.

M functions are not allowed in the G84 block.

Example:



NOTE:
0.Z means Z=0

Program:

```
(UGS,X,-60,60,Y,-70,70)
N90 (DIS,"TAP M8-TRACTION TYPE COMPENSATOR")
N91 S280 F315 T8.8 M6 M13
N92 G84 R7 Z-15
N93 X-51.96 Y-30
N94 X51.96
N95 X Y60
N96 G80 Z50 M5
```

This program is valid for R.H. tapping operations, because M13 is programmed in block N91. For L.H. tapping operations, simply program M14 (or M04) instead of M13 (or M03).

G84 - Tapping Cycle with Transducer

This G84 code operates when a transducer is mounted on the spindle.

Syntax

G84 [*G-codes*] *R1.. [R2..] Z.. [K..] [F..] [auxiliary]*

---

where:

G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Approach dimension (mandatory). Defines the coordinates for rapid positioning on the machining plane when the cycle starts. The R address is followed by the rapid approach value. It can be programmed directly with a decimal number or indirectly with an E parameter.
R1	Defines the rapid approach coordinates and the the fixed return to work.
R2	Defines the rapid return coordinates.
K	Defines the thread tap pitch. The K address is followed by a value.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.
auxiliary	M, S or T programmable auxiliary functions (see G81).

Characteristics:

With a transducer is mounted on the spindle, the G84 code can be programmed as follows:

- by calculating the F feedrate as if there were no spindle on the transducer.
- by using the K thread pitch. In this case the control automatically calculates the feedrate by multiplying K by the spindle speed in rpm.

WARNING

During the tapping cycle the control ignores the CYCLE STOP pushbutton (except in the rapid traverse approach section) and the FEEDRATE OVERRIDE selector (or softkey).
The SPINDLE SPEED OVERRIDE selector must be disabled by the machine logic. To abort the tapping cycle, the "INTP-ABO" logical function may be used (see variable description on Plus Variables Manual).

Example:

```
N90 (DIS,"TAP M8")
N91 S280 T8.8 M6 M3
N92 G84 R7 Z-15 K1
N93 X-51.96 Y-30
N94 X51.96
N95 X Y60
N96 G80 Z50 M5
```

G84 - Rigid tapping cycle with a transducer mounted on the spindle

Syntax

G84 [*G-codes*] **R1**.. [**R2**..] **Z**.. **K**..[*auxiliary*]

where:

G-codes	Other G codes compatible with canned cycle G84 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the rapid approach and working return location points (mandatory).
R2	Defines the rapid return location points.
Z	Defines the end of tapping points (normally Z). It is given by the Z address followed by a value which can be programmed either directly with a decimal number or indirectly with an E parameter.
K	Defines the male threading pitch. It is given by the K address followed by a value.
auxiliary	Other M, S, T functions (see G81).

Characteristics:

To perform the tapping cycle correctly it is necessary to load two machine parameters with the appropriate values, which, as a rule, vary from machine to another
These parameters are called TKG and TAG.
Accordingly, rigid tapping cycles must be preceded by the definition of these two parameters with the values obtained at the machine installation stage.

We recommend creating a paramacro for the loading of these values (for instance, G840), to be recalled only once within the program that performs the rigid tapping operation.
Hence, the paramacro will be structured as follows:

```
TKG=……
TAG=……
```

WARNING

If a reset command (which stops the program underway) is executed, the TKG and TAG parameters are deleted and will have to be assigned again for the correct execution of rigid tapping.

The methods for the determination of the correct values of the parameters are described in the AMP manual (Chapter 1)

TRP (RMS) - Tapping Return Percentage

The TRP command defines the feedrate percentage variation applied in the retract phase of the tapping cycle. This command is normally defined in a program, but can also be used in blocks entered with a keyboard command or by means of a softkey.

Syntax

TRP = *value*

where:

value	can be programmed directly with an integer number or indirectly with an E parameter of the byte type.

Example:

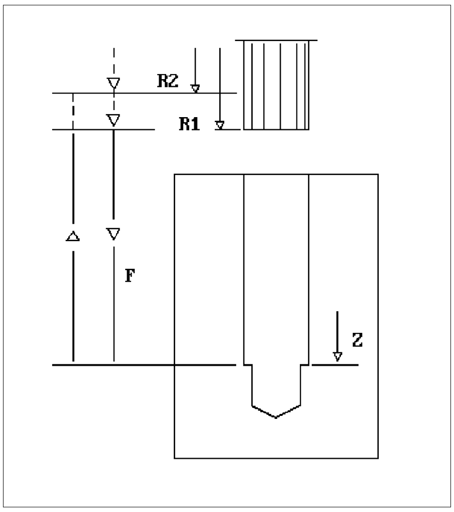
TRP = 110	represents +10% of the programmed F
TRP = 10	represents - 90% of the programmed F

G85 - Reaming Cycle (or Tapping by Tapmatic)

Syntax

G85 [*G-codes*] *R1.. [R2..] Z.. [F..]* [*auxiliary*]

where:



G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the rapid approach coordinates and the the fixed return to work.
R2	Defines the rapid return coordinates.
Z	Defines the hole depth (typically Z). The Z address is followed by the depth value, which can be programmed directly with a decimal number or indirectly with an E parameter.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.
auxiliary	M, S or T programmable auxiliary functions (see G81).

G86 - Boring Cycle

The spindle must belong to a process in exclusive mode.

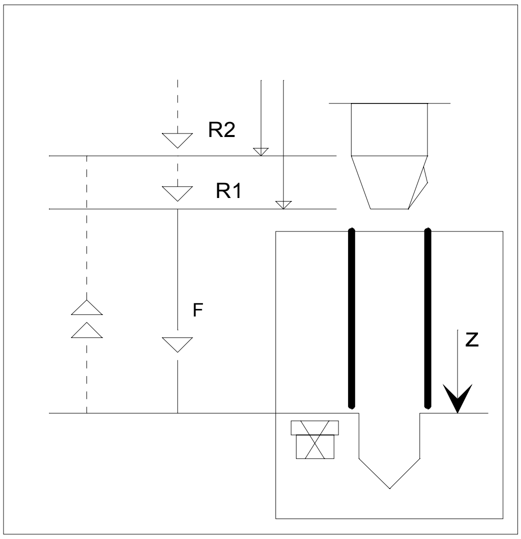
Syntax

G86 [*G-codes] R1.. [R2..] Z.. [F..*] [*auxiliary*]

---

where:

G-codes	Other G codes compatible with G81 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the rapid approach coordinates and the the fixed return to work.
R2	Defines the rapid return coordinates.
Z	Defines the hole depth (typically Z). The Z address is followed by the depth value, which can be programmed directly with a decimal number or indirectly with an E parameter.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.
auxiliary	M, S or T programmable auxiliary functions (see G81).



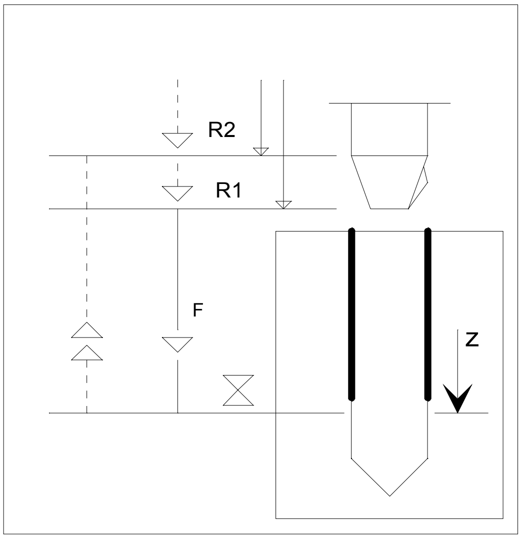
G89 - Boring Cycle with Spot Facing

Syntax

G89 [*G-codes*] **R1**.. [**R2**..] **Z**.. [**F**..] [*auxiliary*]

---

where:

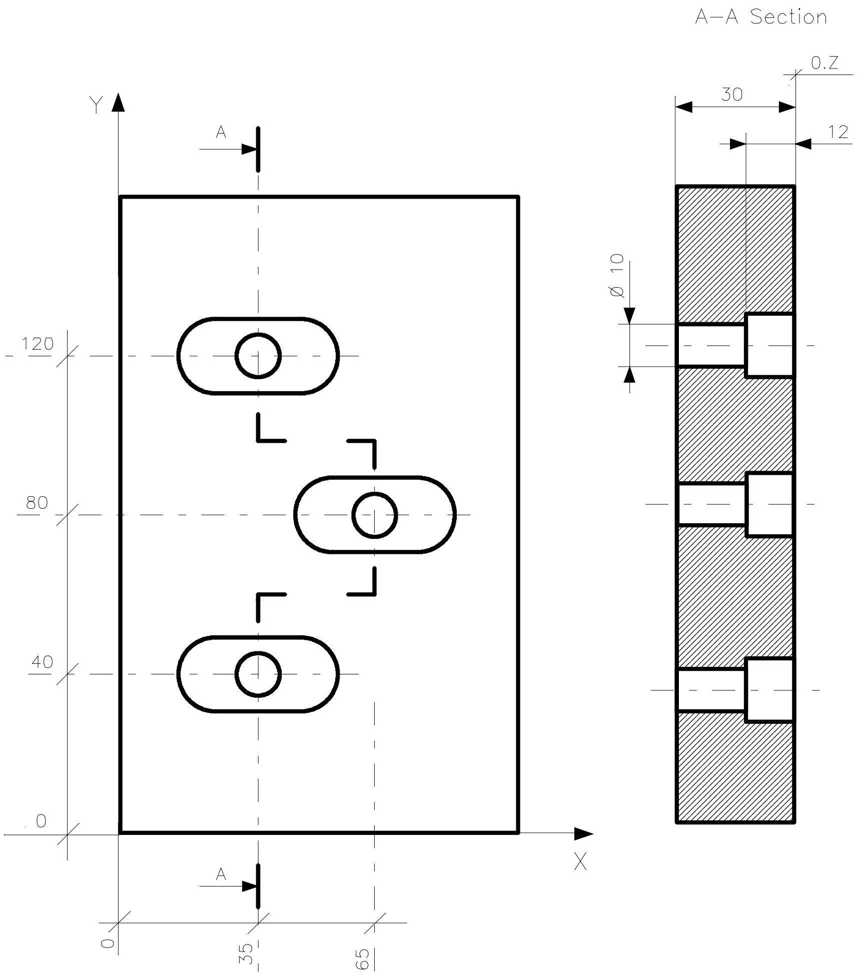


G-codes	Other G codes compatible with G89 (see "Compatible G codes" Table in Chapter 1).
R1	Defines the rapid approach coordinates and the the fixed return to work.
R2	Defines the rapid return coordinates.
Z	Defines the hole depth (typically Z). The Z address is followed by the depth value, which can be programmed directly with a decimal number or indirectly with an E parameter.
F	Defines the feedrate used in the canned cycle operation. It is programmed with the F address followed by the feedrate value.
auxiliary	M, S or T programmable auxiliary functions. In a canned cycle block you can program up to four M functions, one S (spindle speed) and one T (tool selection).

Using two R dimensions in a canned cycle

You can program two R's in a canned cycle when holes must be drilled, tapped, reamed, etc., on the same plane, but are separated by obstacles (clamps, holes on repeated subplanes, etc.)

Example:



NOTE:
0.Z means Z=0

Program:

```
N41 (UGS,X,0,100,Y,0,150)
N42 (DIS,"TWIST DRILL D=10")
N43 S850 F100 T4.4 M6
N44 G81 R-10 R2 Z-36 M3
N45 X35 Y40
N46 X65 Y80
N47 X35 Y120
N48 G80 Z50 M5
```

Updating Canned Cycle Dimensions

When a canned cycle is active in a program, you can program blocks with rapid approach, return and depth coordinates in order to update the cycle moves without re-programming the cycle.

Program blocks whose format is X, Y, R1, R2, Z are performed in this order:

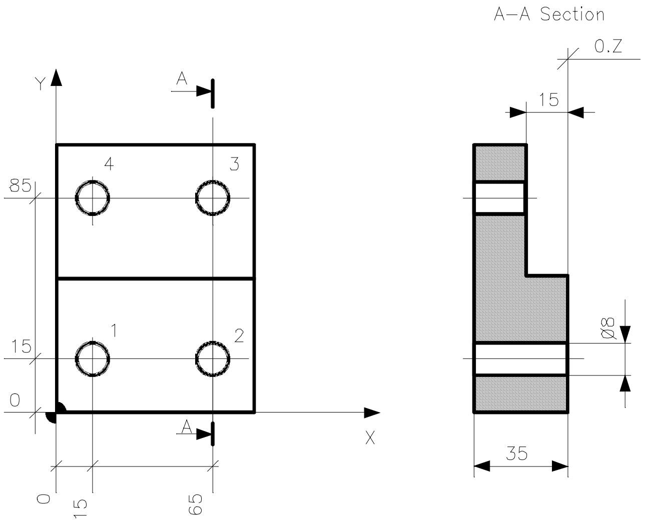
1. X and Y
2. Updated R1 - new rapid approach coordinates
3. Updated Z - new depth
4. Updated R2 - new return coordinates

The table below summarizes the program block formats used for updating canned cycles:

FORMAT	ACTION
X.. Y.. Z..	Performs a canned cycle at XY with new Z depth
X.. Y.. R..	Performs a canned cycle at XY with new R1 rapid plane
X.. Y.. R.. R..	Performs a canned cycle at XY with new (R1) rapid plane and (R2) return dimension
X.. Y.. R.. Z..	Performs a canned cycle at XY with new R1 rapid plane, and Z depth
X.. Y.. R.. R.. Z..	Performs a canned cycle at XY with new R1 rapid plane, R2 return dimension, and new Z depth
R..	Updates the R1 rapid plane and does not the perform canned cycle at the current location
R.. R..	Updates R1 rapid plane and R2 return dimension; does not perform a canned cycle at the current location
R.. Z..	Updates R1 rapid plane and Z depth; does not perform a canned cycle at the current location
R.. R.. Z..	Updates R1 rapid plane, R2 return, and Z depth dimensions; does not perform a canned cycle at the current location

Updating R dimensions (upper limit and lower limit) during execution

Example 1:

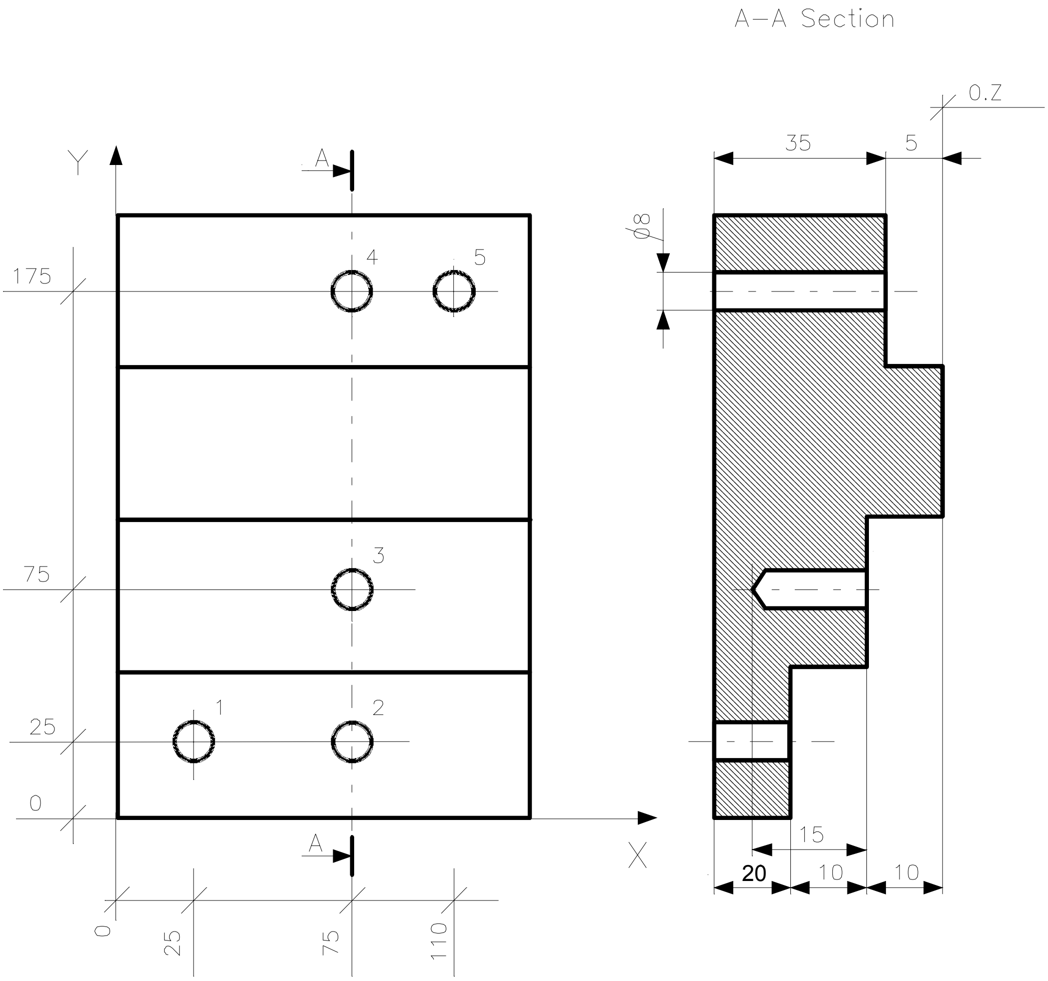


NOTE:
0.Z means Z=0

Program:

```
    N35 (DIS,"TWIST DRILL D=8")
    N36 S1000 F100 T4.4 M6
    N37 G81 R3 Z-42 M3
1   N38 X15 Y15
2   N39 X65
3   N40 Y85 R-13
4   N41 X15
    N42 G80 Z50 M5
```

Example 2:

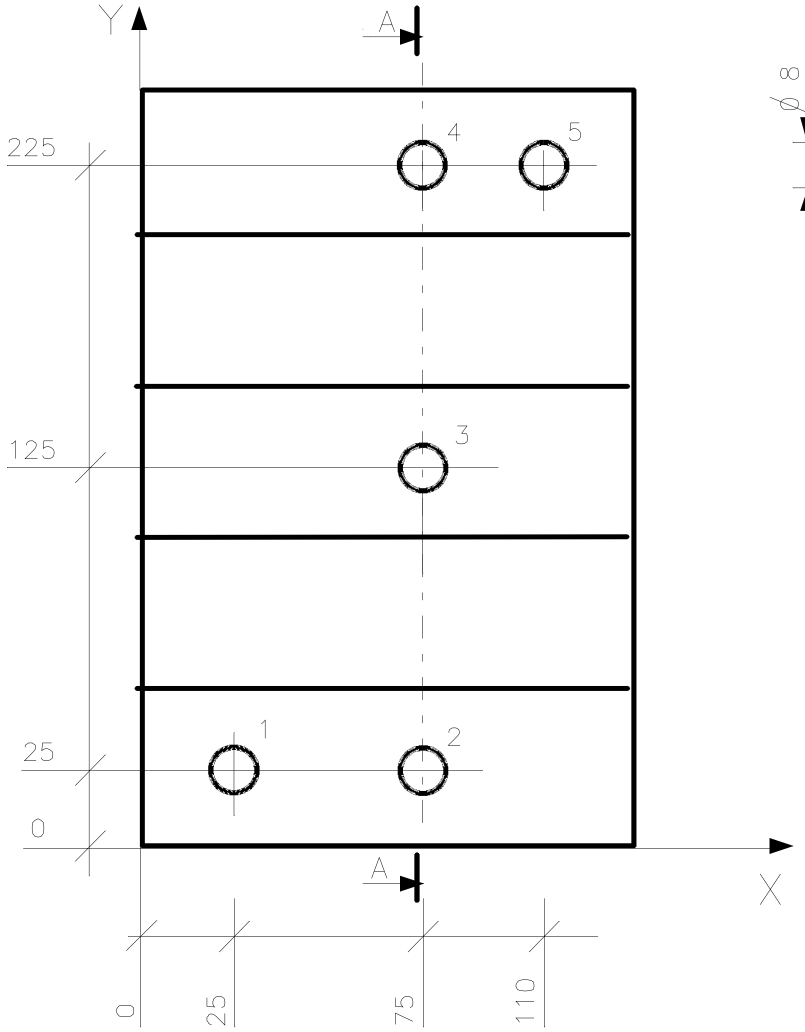


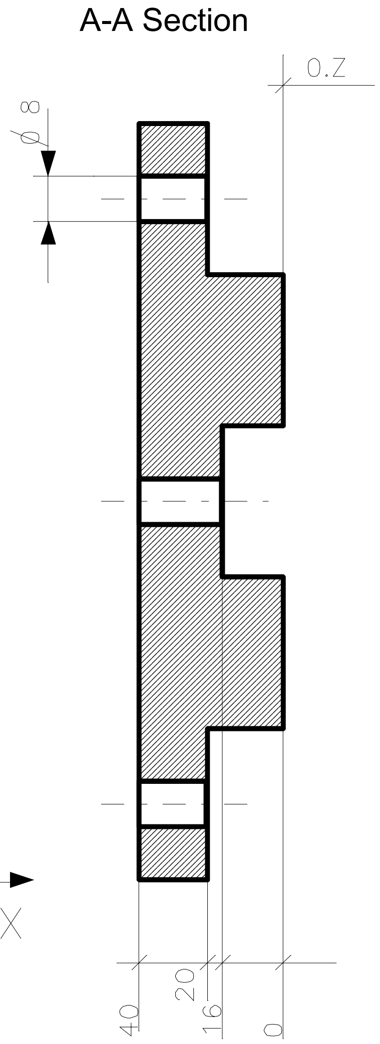
NOTE:
0.Z means Z=0

Program:

```
    (UGS,X,0,100,Y,0,200)
    N42 (DIS,"TWIST DRILL D=8")
    N43 S1000 F100 T5.5 M6
    N44 G81 R-18 Z-46 M13
1   N45 X25 Y25
2   N46 X75 R-18 R-8
3   N47 Y75 R-8 R2 Z-25
4   N48 Y175 R-3 Z-46
5   N49 X110
    N50 G80 Z50 M5
```

Example 3:





NOTE:
0.Z means Z=0

Program:

```
    (UGS,X,0,100,Y,0,200)
    N42 (DIS,"TWIST DRILL D=8")
    N43 S1000 F100 T4.4 M6
    N44 G81 R-18 Z-42 M3 M8
1   N45 X25 Y25
2   N46 X75 R-18 R4
3   N47 Y125 R-14 R4
4   N48 Y225 R-18
5   N49 X110
6   N50 G80 Z2 M5
```

END OF CHAPTER

PARAMACRO

Paramacro Definition

Paramacro subroutines can be used in user-defined cycles. They are called with a 3-digit code.

Modal paramacros are active only in motion blocks that do not include M functions. No other function or code can be programmed when a modal paramacro is active.

Syntax

G*n par-name-1* [*value-1*] . . . [*par-name-n*] [*value-n*] . . . *["string"]*

---

where:

n	Is a number from 300 to 998
par-name-1. . . *par-name-n*	Are letters (refer to the correspondence table)
value-1 . . . *value-n*	Can be a number or an E parameter or a parametric expression.
string	Character string (max. 99)

Characteristics:

There are two groups of paramacros:

- from G300 to G699   non modal paramacros
- from G700 to G998   modal paramacros

Modal paramacros are reset by G999.

The H, HF and HC parameters are used in the paramacros, The system offers 100 H parameters. Some of them can be used in paramacros. H parameters are classed as follows:

- Parameters from H0 to H51 are associated to letters and cannot be used in paramacros.
- Parameters from H52 to H99 can be used for math operations in paramacros.

When H parameters are used in paramacros, all the operations can be calculated with E parameters.

Paramacros allow 4 nesting levels.

When a series of paramacros are nested, the H, HF and HC parameters in a nested paramacro will reset the equivalent H, HF and HC parameters in upper paramacro levels.

Example:
G300 A1 B2 C3 D4

H0 is assigned to 1
H1 is assigned to 2
H2 is assigned to 3
H3 is assigned to 4

HF0 is set
HF1 is set
HF2 is set
HF3 is set

All other HF are reset.

If G300 calls G400 A10 C30, then the value of H0 is 10 and the value of H2 is 30. HF0 and HF2 will be set and all other HF's will be reset.

The values assigned to H0 and H2 in G300 are now lost.

All the other parameter types available with 10 Series can theoretically be used in paramacros. However, to avoid interactions with E parameters used elsewhere, it is preferable to use H parameters where possible.

HC Parameters

If a block calling a paramacro also includes specification of a character string between inverted commas, this string is made available for the paramacro in the HC character array (100 characters).

If no such string is specified, the entire HC array is reset.
If the string was programmed, it is put in the HC variables with a terminator '0' added; for this reason, maximum length of the string is 99, even if the HC array is 100 characters long.

Examples:

```
!PROFILE=...........
G600 "PROFILE" A50 ;call to paramacro G600
. .
. .

;----------------- in the paramacro...........
(DIS,HC0.10)
(CLS,?HC0.10)
```

Situation of the HC parameters after the call to the paramacro:

HC0="P" HC1="R" HC2="O" HC3="F" HC4="I" HC5="L" HC6="E" HC7=0

HO=50 HFO=1

The following table shows the correspondence between letters and H parameters.

LETTER	PARAMETERS H	 HF	LETTER	PARAMETERS H	 HF
A	H0	HF0	a	H26	HF26
B	H1	HF1	b	H27	HF27
C	H2	HF2	c	H28	HF28
D	H3	HF3	d	H29	HF29
E	H4	HF4	e	H30	HF30
F	H5	HF5	f	H31	HF31
G	H6	HF6	g	H32	HF32
H	H7	HF7	h	H33	HF33
I	H8	HF8	i	H34	HF34
J	H9	HF9	j	H35	HF35
K	H10	HF10	k	H36	HF36
L	H11	HF11	l	H37	HF37
M	H12	HF12	m	H38	HF38
N	H13	HF13	n	H39	HF39
O	H14	HF14	o	H40	HF40
P	H15	HF15	p	H41	HF41
Q	H16	HF16	q	H42	HF42
R	H17	HF17	r	H43	HF43
S	H18	HF18	s	H44	HF44
T	H19	HF19	t	H45	HF45
U	H20	HF20	u	H46	HF46
V	H21	HF21	v	H47	HF47
W	H22	HF22	w	H48	HF48
X	H23	HF23	x	H49	HF49
Y	H24	HF24	y	H50	HF50
Z	H25	HF25	z	H51	HF51

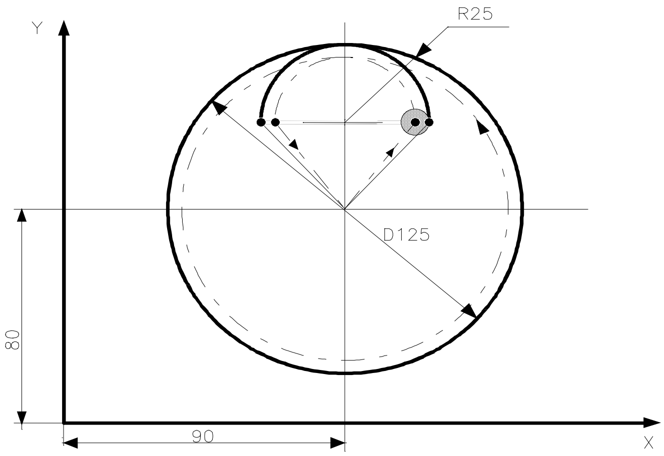
Example 1:
N45 G777 A(E8) R22.5 F(E2) S(E3+5-E1)

E8 is passed to H0
22.5 is passed to H17
E2 is passed to H5

The result of (E3+5-E1) is passed to H18

In this example Boolean parameters HF0, HF17 and HF5 are set to 1.

Example 2:



D=H3=Major diameter
R=H17=Minor radius
F=H5=Feedrate

Program:

This is an example of milling/boring cycle using paramacros.

```
;MAIN PART PROGRAM
    .
    .
    .
N20  G601  D125  R25  F160
N21 . . .
```

```
; PARAMACRO G601
G0 X90 Y80
G92 XY
(GTO,END,HF3=0)
(GTO,END,HF17=0)
(GTO,END,HF5=0)
H57=H3/2
H58=H57-H17
G1 G41 XH17 YH58 F2000
G3 X0 YH57 I0 JH58 FH5
I0 J0
H59=NEG(H17)
G40 XH59 YH58 I0 JH58
G1 X0 Y0 F2000
(GTO,F)
"END"
G99
(DIS,"OMITTED PARAMETERS")
M ..
"F"
```

DAN - Define Axis Name

This command associates the name of a characterised axis to the name used in a paramacro.

Syntax

(DAN,*par-ax1 char-ax1*[**,***par-ax6 char-ax6*]**)**

(DAN)

where:

char-ax1 **...** *char-ax6*	Are the names of up to six characterised axes.
par-ax1 ... par-ax6	Are the names of six axes used in a paramacro.
no parameters	(DAN) without parameters disables (DAN) mode.

Example 1:

(DAN,PX,QY,DZ)	X,Y,Z are replaced by P,Q,D so X,Y,Z are not usable.

Example 2:

(DAN,PX,QY,DZ)	X,Y,Z are replaced by P,Q,D so X,Y,Z are not usable.
(DAN,WA)	A is replaced by W, furthermore X,Y,Z are re-enabled and P,Q,D,A are not now usable. By reprogramming (DAN,...) all previous associations are cancelled and the current ones become active.

IMPORTANT

After the three-letter mnemonic DAN has been used, if the axes concerned are those of the interpolation plane, this plane must be redefined using the new names.

PMS – S as a paramacro
PMT – T as a paramacro
PMM – M as a paramacro

These commands enable and disable the management of the relative ancillary functions S, T, M as paramacros.
For further information on how to program these paramacros see the paragraph on the “S, T, M ancillary functions executed as paramacros” in this chapter.

Syntax

PMS=value
PMT=value
PMM=value

---

where:

value	This value can be:
	0 Disables the management of the S, T, M functions as paramacros – these functions are executed in standard mode
	1 Enables the management of the S, T, M functions as paramacros – these functions associate the execution of the corresponding paramacro

NOTE:

The values ascribed to the PMS, PMT and PMM variables can be changed as follows:

- Through AMP at the configuration stage
- Through a part program with the syntax described

The characterisation values are restored with a reset.

S, T, M ANCILLARY FUNCTIONS EXECUTED AS PARAMACROS

The execution of the S T M ancillary functions can be associated with paramacro type subprograms, i.e. in the presence of a block containing at least one S, T and/or M function, previously configured as a paramacro, the corresponding subprogram is executed, according to the criteria currently defined for paramacros.
Syntax:

same as the syntax used for ancillary functions executed in standard mode.

ANCILLARY CODE	PARAMACRO
S	In the presence of an “Sx” function, e.g. on process “n”, paramacro “P0nS” is executed with parameter H18 = x
T	In the presence of a “Tx” function, e.g. on process “n”, paramacro “P0nT” is executed with parameter H19 = x
M	In the presence of an “Mx” function, e.g. on process “n”, paramacro “P0nM00x” is executed with parameter H12 = x

Characteristics:

- The functions generating paramacro calls need not be programmed at the start of the block.
- If M, S and/or T are programmed simultaneously, the name of the procedure to be called is determined by the first function which is configured as a paramacro, and any other functions programmed in the block are called with the respective parameters H, HF and HC, as described in the programming manual (chapter 9). For example, if both M and T are configured as paramacros and you program TxMy, this will generate a call to macro P0nT whilst the programming of MyTx will call the "P0nM00x" macro, with parameters H19=x and H12=y in either case.
- For the execution of the paramacro in MDI mode, see paragraph “Executing blocks entered from the keyboard” in the user manual.
- At the end of the paramacro, the execution mode (MDI, AUTO, STEP) in which the system was working before the call is restored.
- The type of paramacro being executed can be read in the **SW21** variable and it is:
  - 1000 for a paramacro called by the S function
  - 2000 for a paramacro called by the T function
  - 3xxx for a paramacro called by the Mxxx function

- The *pathname* to be used to call the paramacros can also be specified with the PTH triliterals described below in this chapter. If the pathname is omitted, the search for the directory containing the subroutine is performed as follows:

  1. case: no pathname has been specified through a PTH instruction.
  The system searches for the paramacro in the directory containing the calling program and, if it does not find it, searches for it in the DOS directories to which logic names have been associated during the machine characterisation process.

  2. case: a pathname has been specified through a PTH instruction.
  The system searches for the paramacro in the directory specified with a PTH and, if it does not find it, searches for it in the DOS directories to which logic names have been associated.

Configuration: three process variables (PMM, PMS, PMT, defined previously), if set on 1/0, enable/disable the execution of the corresponding functions as paramacros at a general level.
Moreover, the execution of ancillary function M as a paramacro is identified by an ad hoc bit, defined in AMP.

Hence, if PMM = 1 and the “Mx” function is:

- configured in the “n” process but its bit = 0, the paramacro is not executed (standard Mx is executed instead)

- configured in the “n” process and its bit = 1, the “P0nM00x” paramacro is executed

- not configured in the “n” process, an “M not defined” error message is displayed

Example:

Execution of the block described below in process 1, in which the auxiliary function M2 is configured with the paramacro bit enabled

N20 X10 M2 T1.0 S2000

PMM	PMT	PMS	Paramacro name	Parameters
1	1 or 0	1 or 0	P01M002	H23 = 10; H12 = 2; H19 = 1.0 H18 = 2000
0	1	1 or 0	P01T	H23 = 10; H12 = 2; H19 = 1.0 H18 = 2000
0	0	1	P01S	H23 = 10; H12 = 2; H19 = 1.0 H18 = 2000

This is an example of how the relative paramacros are developed:

```
;PARAMACRO P01M002
E0 = H12
PMM = 0
ME0
"TEST_T"
(GTO,TEST_S,HF19=0)
E1 = H19
PMT = 0
TE1
"TEST_S"
(GTO,END,HF18=0)
E2 = H18
PMS = 0
SE2
"END"
```

```
;PARAMACRO P01T
E1 = H19
PMT = 0
TE1
"TEST_M"
(GTO,TEST_S,HF12=0)
E0 = H12
PMM = 0
ME0
"TEST_S"
(GTO,END,HF18=0)
E2 = H18
PMS = 0
SE2
"END"
```

```
;PARAMACRO P01S
E2 = H18
PMS = 0
SE2
"TEST_M"
(GTO,TEST_T,HF12=0)
E0 = H12
PMM = 0
ME0
"TEST_T"
(GTO,END,HF19=0)
E1 = H19
PMT = 0
TE1
"END"
```

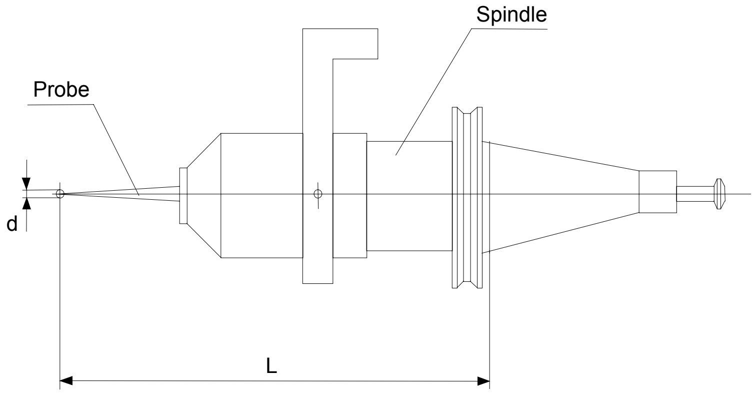
END OF CHAPTER

Chapter 10

PROBING CYCLES

MANAGING AN ELECTRONIC PROBE

An electronic probe is a measuring device that can be mounted on the spindle and controlled like a tool. The probe can also have length and diameter offsets associated with it. Besides, it can be mounted in a fixed position and used as an electronic gauge to requalify the tool length.
The figure that follows shows an electronic probe with its relevant dimensions.



Electronic Probe

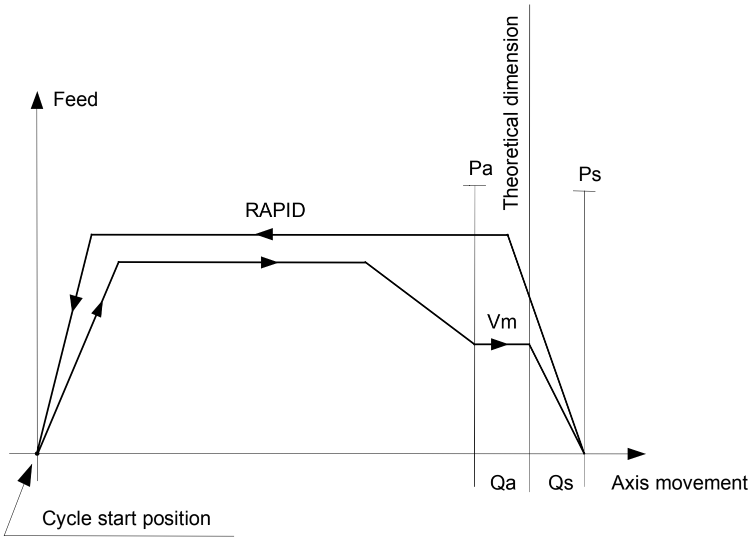
Some G codes perform specific measuring or probing cycles when an electronic probe is mounted on the spindle.

G CODE	FUNCTION
G72	Measures the co-ordinates of a point
G73	Measures the co-ordinates of the circle center and radius
G74	Measures variances from theoretical points (for probes mounted in a fixed position rather than on the spindle).

The control stores measured values in E parameters that you define in the probing cycles (G72-G74).
When the control executes a probing cycle, it measures a point through the following sequence of moves:

1. Rapid to the approach point (Pa).
2. Move at measuring speed (Vm) to the point where the probe triggers, then stop and store the dimensions. If the probe does not trigger, move only as far as the end safety point (Ps).
3. Rapid return to the start position of the probing cycle (hole center in G73). The execution of that phase depends on an (AMP) machine characterisation parameter or the corresponding DPP three-letter block mode parameter.

The diagram below shows moves and feed of a probing cycle.



The following errors may occur during a probing cycle:

- If the probe does not trigger before reaching the safety point, it returns to the start point of the cycle. The control panel displays a message and the system goes into an error condition.
- If the probe does not reset properly on the way back to the start point after a successful probing cycle, a message is displayed and the system goes on error condition.
- If the probe is carried out during the rapid approach phase, it returns to the start point of the cycle. The control panel displays a message and the system goes into an error condition.

NOTE:
If the probing cycle is executed with G27 or G28 active, both the approaching movement and probing are carried out in continuous mode, while the stopping movement obtained after the probe input has been found is executed in point to point mode.

WARNING

Values obtained from probing cycles are always expressed in the characterised unit (AMP). Therefore they are not affected by G70/G71 functions.

PRESETTING A PROBING CYCLE

The first time you use a probe, or whenever probing cycle conditions change, you must:

1. define the probing parameters
2. dynamically measure the diameter of the probe ball
3. requalify the probe with respect to the spindle axis
4. dynamically measure the length of the probe.

DPP (DPT) - Defining Probing Parameters

Use the DPP command to define the probing parameters from the keyboard or in a program. These parameters can also be defined on the control with a specific data entry via softkey (MACHINE SET-UP and PROBE SET-UP soft-keys). When these are not defined the system uses the values defined in AMP.

Syntax

(**DPP**,*approach*,*safety*,*speed*[*, mode*])

---

where:

approach	Is the approach tolerance expressed in mm or inches.
safety	Is the safety tolerance expressed in mm or inches.
speed	Is the measuring speed expressed in mm/min or inches/min.
mode	Is the value which indicates whether to perform the rapid return phase at the end of the probe. It can be one of the following values : 0 = perform the return phase 1 = do not perform the return phase If no parameter is present then the default 0 is assumed.

Example:
Instruction used from the program
(DPP,10,12,1000)

WARNING

"Approach", "speed" and "safety" values are displayed in the unit of measurement that is active when the display is requested (G70 or G71).

Dynamic Measurement of the Ball Diameter

To dynamically measure the apparent diameter of the probe ball, use a requalification ring, cylinder, or similar device with a center assumed to be the absolute origin 9 of axes X and Y.

Probe Requalification

Typically, before using the probe, the center of the probe ball is not centered on the spindle axis. To center the probe ball on the spindle axis, you must requalify the probe.
Use the hole of the requalifying ring, cylinder, or similar device for this operation. The center of the hole is assumed to be origin 9 for axes X and Y.

Dynamic Measurement of the Probe Length

To measure the length of the probe dynamically, use the requalification ring, cylinder, or similar device whose reference surface (top) is assumed to be origin 9 for the length axis Z.

Probe Presetting

The following steps must be executed to preset the probe:

1. Use a requalification ring for reference.
2. Mount the requalification ring on the B rotary axis with its front surface at absolute origin 9 for the B rotary axis.
3. Position absolute origin 9 for axes X and Y at the center of the ring hole.
4. Position absolute origin 9 for the Z axis on the top surface of the ring.

The initial values stored in the probe offset are:

length1 = nominal length of the probe with respect to the axis of the ball.

Diameter = 0

Example:

```
N1  (DIS, "DPP, UPA, UPO, ")
N2 T30.30 M6        - probe on spindle
N3  (UAO, 9)
N4  (DPP, 10, 12, 600)  - defines probing parameters
N5  UPA=0
N6  UPO=0
N7  E30=...         - assigns diameter to the hole of the sample ring
N8  E31=E30/2
N9  E32=...         - assigns distance from Z=0 to probing surface on the  Z axis (usually = 0)
N10  E33=E31+10
N11  GB0            - only if the ring is mounted on the indexing table
N12  XY
N13  Z-4
N14  G73 rE31 E40       - measures hole co-ordinates (center and radius)
N15  Z100
N16  (DIS, "UPA=", E40, "UPO=", E41)
N17  M0
N18  UPA=E40        - requalifies probe abscissa
N19  UPO=E41        - requalifies probe ordinate
N20  E34=(E30-E42*2)            - diameter of apparent ball
N21  (DIS, "DIAMETER=", E34)
N22  M0
N23  (RQP, 30, 30, dE34)        - stored ball diameter on d offset
N24  T30.30 M6      - enables new offset
N25  GXYE33
N26  G72 ZE32 E43       - measures Z dimension on ring surface
N27  E35=E43-E32        - variance between nominal and real value
N28  Z100
N29  (DIS "VARIANCE.Z=", E35)
N30  M0
N31  (RQP, 30, 30, LE35)        - requalifies Z length offset
N32  M30
```

PROBING CYCLES

These are G codes that define probing cycles:

G72	Point measurement (with probe ball diameter compensation)
G73	Measurement of hole parameters
G74	Point measurement (without probe ball diameter compensation)

In the cycles G72, G73 and G74, E parameters are coupled with the axes in the order in which the latter are characterised (AMP) and displayed, and not in the order in which they are programmed.

With probing on rotated plane virtual axes, the E parameters are matched to the virtual axes in the order in which they have been defined in the three-letter code UPR.

G72 - Point Measurement with Compensation

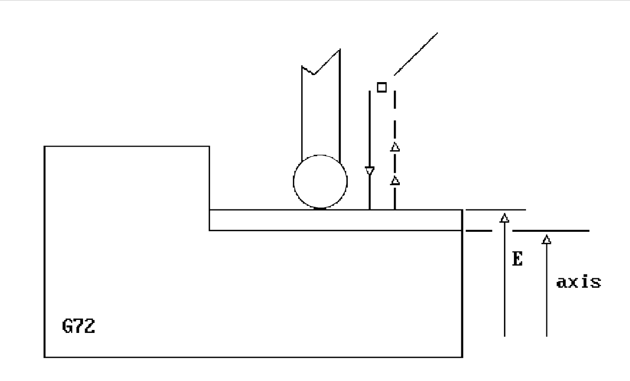
G72 lets you use linear moves and a probe to measure the co-ordinates of a point.

Syntax

G72 *axis1* [*axis2*] [*axis3*] *E-par*

---

where:



axis1 . . . axis3	Are the axes that will move during the probing cycle. The dimensions programmed in the G72 block are the theoretical positions where the probe is expected to impact the target point. You can program up to three axes and dimensions in the G72 block.
E-par	Defines the E(x) parameter for storing the measured dimension of the first axis, in order of configuration, programmed in G72; the dimensions of other axes are stored sequentially in order of configuration in E(x+1) , E(x+2) and E (x+3) in virtual axes probing.

Characteristics:

The co-ordinates measured by the probe are stored in E parameters that are defined by the G72 cycle. The control stores the co-ordinates of the axes beginning with the E parameter specified in the cycle. Measures are taken applying cutter diameter compensation to the probe. To ensure high accuracy the surface must be perpendicular to the measuring move.

The probe can also be performed on virtual axes activated by the UPR function or on the axis which identifies the direction of the activated tool (TCP, 5).

When the probe is performed on the virtual axes defined with UPR, the values measured in relation to the virtual axes are stored in E (x), E (x+1), E (x+2).

When the probe is performed on the tool directional axes, the system stores the value of dimensions measured in relation to those axes in E (x) and the values measured, in relation to the part of the Cartesian linear axes defined in the TCP table or corresponding virtual axes if UPR virtual mode is active in E (x+1), E (x+2), E (x+3).

Example:
The axes are configured in AMP in the following order: XYZ

```
G72 X100 Y50 E32
G72 Y50 X100 E32
```

In both cases the values that the control calculates for X and Y are stored sequentially in E32 and E33.

Example:
Probing on plane rotated with virtual axes.

```
(UPR,0,XYZ,UWV,30,25,0)
G72 U25 V-10 W-25 E30
```

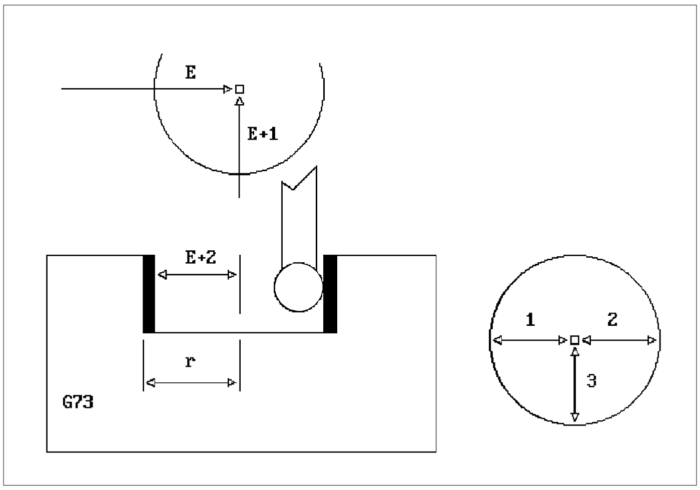
The dimensions of the system of three virtual axes are stored in E30, E31 and E32 according to the order defined in the UPR command or U in E30, W in 31 and V in 32.

G73 - Hole Probing Cycle

G73 lets you use a probe to measure the dimensions of a hole on the active interpolation plane.

Syntax

G73 *r.. E-par*



where:

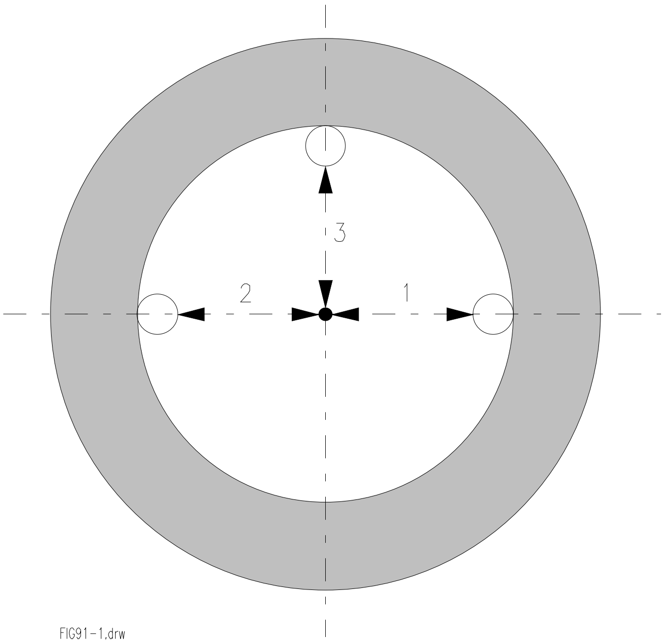
r	Defines the theoretical hole radius. r is followed by the radius length.
E-par	Is the first E parameter in which the system starts storing values: first E parameter: center abscissa (typically X) second E parameter: center ordinate (typically Y) third E parameter: radius

Characteristics:

If you program only one E parameter, the three values detected during the probing operation (circle center co-ordinates and radius) are stored in E parameters sequentially starting from the specified E parameter.

The probe can also be performed on virtual axes activated by the UPR function. In this case the values measured for the centre in relation to the virtual plane are stored in E (x) and E (x+1) and the radius is stored in E (x+2).

Before activating the G73 cycle, the axes of the machine tool must be positioned on the hole center. Measures are taken applying cutter diameter compensation to the probe.



Probing moves

Example:
G73 r100 E55 probing moves

The co-ordinates of the circle center (abscissa and ordinate) and the actual radius are stored in E55, E56, and E57 respectively.

G74 - Tool Requalification Cycle

G74 is a tool requalification cycle. G74 uses a fixed probe (such as an electronic gauge) to measure variances from theoretical points with the tool mounted on the spindle.

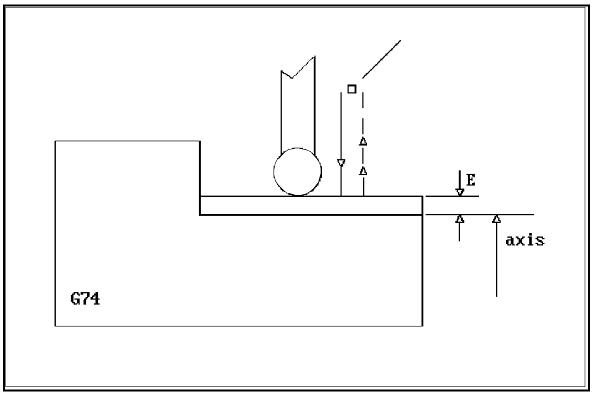
Syntax

G74 *axis1* [*axis2*] [*axis3*] *E-par*

---

where:

axis1. . . axis3	Are up to three simultaneous axes.
E-par	Defines the E(x) parameter for storing the deviation from the point measured and the point programmed for the first axis programmed in G74. The deviation values of the other axes, if programmed, are stored sequentially in E(x+1) and E(x+2).



Characteristics:

G74 can be used for requalifying a tool or checking tool wear. The calculation of measured dimensions does not take into account the tool offset, since the cycle is checking the actual "tool" dimension.

The steps in the G74 cycle are similar to those in the G72 cycle. The difference between both cycles is how the control executes calculations based on measured dimensions. The control does not consider the diameter of the probe ball and stores the variance from theoretical dimensions in the parameters specified in the G74 block.

The probe can also be performed on virtual axes activated by the UPR function or on the axis which identifies the direction of the activated tool (TCP, 5).

When the probe is performed on the virtual axes defined with UPR, the deviation values between the points measured and the programmed points in relation to the virtual axes are stored in E (x), E (x+1) and E (x+2).

When the probe is performed on the tool directional axes, the system stores the deviation value between the point measured and the programmed point in relation to that axis in E (x) and the values measured, in relation to the part of the Cartesian linear axes defined in the TCP table or corresponding virtual axes if the UPR virtual mode is active in E (x+1), E (x+2) and E (x+3) .

Example:
G74 X60 E41

E41 is given by the formula: E41 = Pm - Pt

where: Pm is the measured point
Pt is the theoretical point

UPA (RTA) - Update Probe Abscissa

Defines the probe requalification value for the abscissa (typically X).

Syntax

UPA=*value*

---

where:

value	is the abscissa requalification value expressed in millimetres.

UPO (RTO) - Update Probe Ordinate

Defines the probe requalification value for the ordinate (typically Y).

Syntax

UPO=*value*

---

where:

value	is the ordinate requalification value expressed in millimetres.

ERR - Managing Probing Errors

Cycles G72, G73 and G74 enable you to handle process errors either automatically or from part program. You can choose the method by setting the ERR parameter.

For further information on probing error management see Appendix C

OPERATIONS WITH A NON-FIXED PROBE

With probing cycles G72 and G73 you can:

- Requalify origins by:
  - probing reference surfaces
  - centring on a hole
- Check the dimensions of:
  - diameters
  - planes and depth of holes.

Requalifying Origins by Probing Reference Surfaces

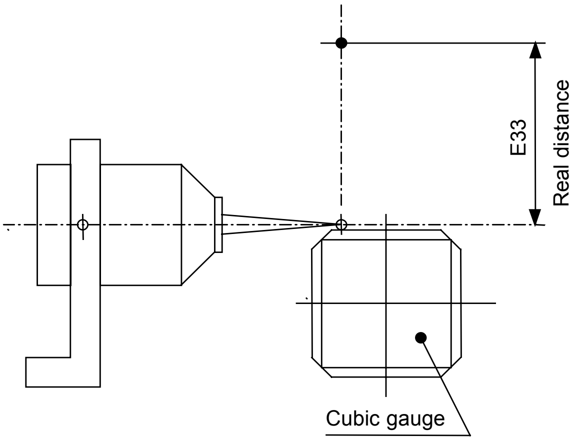
Origins may change due to:

- Changes in temperature (thermal drift)
- New pallet used for machining

These changes may require that you to requalify the system origins by probing a reference surface.

Thermal drift

This procedure uses a requalifying cube that is placed at a precise location on the machine.



Main Program:

```
    .
    .
    .
N99 E33=-300
    .
    .
    .
/N100 (CLS,TEST3)
```

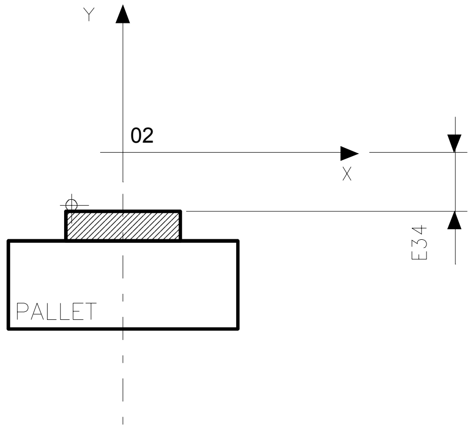
Subprogram TEST3:

```
N500 (DIS,"RQO-DT")
N501 G72 Y(E33) E32         ;Measured distance stored in E32
N502 E32=E32-E33
N503 (RQO,1,Y(E32))         ;Requalify origin 1 for Y axis
N504 (RQO,2,Y(E32))         ;Requalify origin 2 for Y axis
N505 (RQO,3,Y(E32))         ;Requalify origin 3 for Y axis
```

New pallet for machining

Main Program:

```
M . . . ;                   ;Pallet Change
N199 T30.30M6;              ;Probe on spindle
N200 (UAO,2)
N201 GXY
N202 E10=2
N203 E34=-250
/N204 (CLS,TEST4)
```



TEST4 subroutine:

```
N500 G72 Y(E34) E30
N501 E31=E30-E34
N502 (RQO,E10,Y(E31))       ;Requalify origin 2 for Y axis
```

Requalifying Origins by Centring on a Hole

Before you perform this requalification cycle, you must program the X and Y axes to position on the hole axis and program the probe to position into the hole.

Example:

```
N200 (DIS,"REQUALIFYING ORIGIN FOR X AND Y AXES")
N201 T11.11 M6
N202 GX180 Y60
N203 Z-130
N204 G73 r50 E35             ;Measuring cycle diameter 100. X and Y coordinates are
N205 E35=E35-180             ;stored in parameters E35 and E36
N206 E36=E36-60
N207 (RQO,1,X(E35),Y(E36))   ;Requalifies origin 1 for X (E35) and Y (E36)
```

Checking Diameters

You can use a probe to check the hole diameters. With proper programming you can compare detected values with allowed values, and decide whether to continue machining or branch to a block containing a label.

In the following program, the control measures the deviation between the actual and the theoretical diameter of a hole and compares this deviation with the tolerances. The comparison may have three different results and courses of actions:

- Hole diameter is within the allowed tolerance - the program continues.
- Hole diameter is greater than the allowed tolerances - tool is automatically timed out of use (label A3). The program stops (M00) and the part is rejected.
- Hole diameter is less than the allowed tolerances - tool is automatically timed out of use (label A4). The program branches to label A1, where the hole reaming cycle is repeated using the alternative tool that is specified in the tool life file.

Example:
The following is an example of diameter checking.
Hole diameter and tolerance: D=100 +0.02/-0.015

Program:

```
"A1" N111
N112 GZ-150
N113 (DIS,"REAMING HOLE D=100")
N114 F..S..T13.13 M6
N115 GX-120 Y80 M13
    .
    .
    .
N129 (DIS,"HOLE TOLERANCE CHECK D=100")
N130 T14.14 M6                              ;probe in spindle
N131 GX-120 Y80                             ;XY positioning to hole center
N132 Z-85                                   ;Z positioning
N133 G73 r50 E30                            ;hole radius stored in E32
N134 E32=E32*2
N135 (DIS,E32)
N136 (GTO,A3,E32>100.02)
N137 (GTO,A4,E32<99.985)
N138 GZ150
N139 (DIS,"WORKPIECE IN TOLERANCE")
    .
    .
    .
N2100 M30
"A3" N2101 (DIS, "Hole too big")
N2102 M00
"A4" N2103 (DIS, "Hole too small")
/N2104 (GTO,A1)
```

Checking Plane Dimensions and Hole Depths

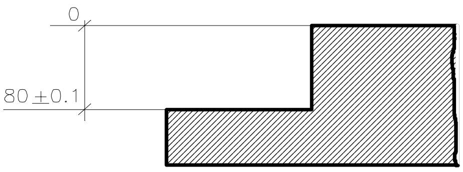
You can use a probe to check the dimensions of planes and the depths of holes. With proper programming you can compare detected values with allowed values, and decide whether to continue machining or branch to a block containing a label.

In the following programming example, the control detects the deviation of actual dimensions from programmed dimensions. It compares this deviation with allowed tolerances. The comparison has three possible results:

- Continue machining
- Stop machining and reject the part because the dimension is too long
- Requalify the tool and repeat the program from label C1 because the dimension is too short

Example:
The following example check of the dimensions of a plane
Theoretical dimension, with respect to Z axis zero: - 80 ± 0.1

Program:



```
    .
    .
    .
"C1" N252
N253 (DIS," . . .")
N254 F. . S. . T23.23 M6
    .
    .
    .
N267 (DIS,"CHECK PLANE DIMENSION 80")
N268  T30.30 M6                    ;Probe on spindle
N269 GX150 Y35                     ;XY positioning on the point to be checked
N270 G72 Z-80 E30                  ;Distance measured and stored in E30
N271 (GTO,C2,E30 <-80.1)           ;Branch to C2 if E30 < min coordinate
N272 (GTO,C3,E30>-79.9)            ;Branch to C3 if E30 > max coordinate
N273 (DIS,"WORKPIECE IN TOLERANCE")
    .
    .
    .

N2100 M30
"C2" N2101 (DIS,E30)               ;Dimension too long
N2102 M00
"C3" N2103 (DIS,E30)               ;Dimension too short
N2104 E31=-80-E30                  ;Variance between virtual and actual dimension
N2105 (RQT,23,23,LE31)             ;Requalify length offset 23 (Z)
N2106 (GTO,C1)
```

OPERATIONS THAT USE A FIXED PROBE

When you use the G74 probing cycle with a probe that is fixed in position (like an electronic gauge) and a tool mounted on the spindle, you can:

- requalify tools (automatic tool offset modifications)
- check tool wear.

Example 1:
This is an example of tool length offset requalification:

```
N170 G X100 Y100              ;Positioning on the measuring point (probe position)
N180 G74 Z-50 E30             ;Deviation measured and value stored in E30
N190 (RQT,10,1,LE30)          ;Requalification of offset 1 for tool 10 in length (Z) by E30
```

Example 2:
This is an example of tool length offset and tool diameter offset modification.

```
N200 G X100 Y100              ;Positioning on the measuring point (probe position)
N210 G74 Z-50 E30             ;Measured deviation (Z axis and value stored in E30)
N220 G X150
N230 Z-60
N240 G74 X130 E31             ;Measured deviation (X axis) and value stored in E31
N250 E31=E31*2
N260 (RQT,10,1,LE30,dE31)     ;Requalify offset 1 of tool 10 in length (Z) by E30 and in
                               diameter (d) by E31
```

Example 3:
This is an example of tool wear inspection with probe error management (Error management methods are described in Appendix C):

```
N480 (DIS,"TOOL D=10")
N490 T10.10M6
. . . . .
. . . . .
N600 (DPP,10,5,500)
N610 (UAO,9)
N620 GXY
N630 ERR=1                     ;Error managed by program
N640 G74 Z0 E35                ;Tool length measurement
N650 (GTO,A2,STE=1)            ;Branch to A2 if tool is broken (no point probed within the
                               ;5 mm safety distance)
N660 ERR=0                     ;Error managed by system
. . . . .
N1500 M30
"A2" N2001 (TOU,10)            ;Declares tool 10 out of use
N2002 (DIS, "TOOL K.O.")
N2003 M0
```

END OF CHAPTER

MANAGING THE SCREEN

GRAPHICS VISUALIZATION

This chapter discusses a class of commands that let you control visualisation of graphics and variables from part program. These commands are listed in the table below.

COMMAND	FUNCTION
UGS	Enables a graphic scale
CGS	Clears the graphic scale
DGS	Disables the graphic scale
DIS	Displays a variable

UGS (UCG) - Use Graphic Scale (Machine plot)

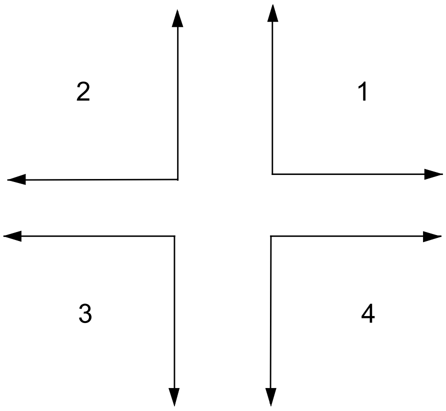
The UGS command initialises the graphic display and establishes the limits and the orientation of the graphic display.

Syntax

(UGS [,*ax-orient*],*abs-axis*,*val1*,*val2*,*ord-axis*,*val3*,*val4* [,*third-axis*]**)**

where:

ax-*orient*	Is a number (from 1 to 4) that selects the type of axis orientation (see figure). The default value is 1
abs-axis	Is the name of the abscissa on the display
val1	Is the lower limit of the abscissa
val2	Is the upper limit of the abscissa
ord-axis	Is a name of the ordinate on the display
val3	Is the lower limit of the ordinate
val4	Is the upper limit of the ordinate
third-axis	Is the name of the third axis (generally a spindle axis).



Axes Orientation

Example:
(UGS,1,X,100,150,Y,50,250,Z)

The graphic display shows movements between X100 and X150 on the abscissa and between Y50 and Y250 on the ordinate referred to the current origin.

UGS (UCG) - Use 3D Graphic Scale

Syntax

(UGS ,5,*axis1,val1,val2,axis2,val3,val4,axis3,val5,val6* **[**α,β**])**

---

where:

5	Selects 3D graphic scale
axis1,axis2,axis3	Are the names of the three axes to be displayed
val1,val2	Lower and upper limit of the first axis
val3,val4	Lower and upper limit of the second axis
α	α angular parameter It is the rotation angle to be applied to the horizontal plane during the 3D the display. The typical horizontal plane is XY.
β	β angular parameter. It is the rotation angle to be applied to the vertical plane during 3D display. Typical vertical planes are XY or XZ.

NOTE:
α and β parameters are optional. If they are omitted , the system will take by default:

α **= 30°**

β **= 30°**

CGS (CLG) - Clear Graphic Screen

The CGS command clears the profile from the screen leaving the system of coordinates.

Syntax

(CGS)

DGS (DCG) - Disable Graphic Scale

The DGS command disables the graphic display, deletes the displayed profile, and removes the system of coordinates from the screen. After using the DGS command you need to use another UGS command to reinitialise the graphic display.

Syntax

(DGS)

DIS - Displaying a Variable

The DIS command allows values to be displayed to the operator. The control will show the value in the screen area that is reserved for communications with the operator.

Syntax

(DIS,*operand* **[,***operand* **] [,***operand***] [,***operand***] [,***operand* **])**

where:

operand	It can be a number, a variable or an ASCII string. Up to five operands can be displayed. All 5 operands cannot exceed be more than 80 characters long. If *operand* is a number it is within the normal range for variables (5.5 format). If *operand* is a variable it can be any variable used in assignment blocks. If *operand* is an ASCII string, it can be a message for the operator. The message can be up to 80 ASCII characters long. Program the message between quotes (" ") in the DIS command.

Examples:

(DIS,100)	displays the value 100
(DIS,E27)	displays E27 current value on the screen
(DIS,MSA)	displays current MSA value (machining allowance)
(DIS,"THIS IS AN EXAMPLE")	displays the following string: THIS IS AN EXAMPLE

END OF CHAPTER

Chapter 12

MODIFYING THE PROGRAM EXECUTION SEQUENCE

GENERAL

This chapter discusses commands that let you modify the program execution sequence by:

- repeating part programs
- executing subprograms
- modify the flow of the program
- delaying and disabling
- releasing the program or suspending its execution
- define devices for access operations on files

Block repetition commands

These commands allow a specified set of program blocks to be executed several times. They can be used for repetitious machining operations such as drilling multiple holes.

COMMAND	FUNCTION
RPT	Opens the set of program blocks to be repeated
ERP	Closes the set of program blocks to be repeated

Commands for executing subprograms

The commands in this class are :

COMMAND	FUNCTION
CLS,CLD,CLT	Calls a subroutine to perform
PTH	Defines the default pathname for the subroutine

A subroutine is a series of blocks defining a machining cycle. The subroutine is stored as a separate file with its own name. Control passes to blocks of the subroutine each time that they are called by the CLS command. The subroutine may be called at any time and from any part of the main program.

Commands for modifying program flow

The commands in this class are:

COMMAND	FUNCTION
EPP	Executes a section of a part-program delimited by two labels
EPB	Executes a block of a part-program
GTO	Performs a jump or skip during the execution of the program
IF ELSE ENDIF	Executes sessions of a part-program depending on certain conditions

The GTO command makes the execution of a program jump to a block which contains a specific label. A jump may be unconditional or conditional based on E parameters, machine logic signals or numeric values.
A conditional jump is performed only if the result is true. No jump is performed if the condition is false.
The commands IF, ELSE, ENDIF allow the flow of the program to be modified conditionally without the need to define labels and program skips.

Commands for delaying program execution and disabling blocks

The commands of this class are :

COMMAND	FUNCTION
DLY	Causes a delay in the execution of the program
DSB	Disables slashed blocks

The delay commands can be used to delay the execution of the program for reasons due to synchronisation.

Commands for the releasing or suspending the execution of a program

The commands in this class are :

COMMAND	FUNCTION
REL	Releases the part-program
WOS	Puts the program in wait mode until a signal is received

Commands for device definition

The commands in this class are:

COMMAND	FUNCTION
GDV	Defines the remote device or drive A for file access operations
RDV	Releases the device defined with GDV

These commands define the remote devices or drive A and are used when read/write operations are performed on the file via the language ASSET.

COMMAND FOR PROGRAM BLOCKS REPETITION

RPT - ERP

The RPT and ERP commands define a set of part program blocks that must be executed a specified number of times. The set of blocks begins with the RPT command and ends with the ERP command.

Syntax

(RPT,*n*)

.
.

blocks

.
.

(ERP)

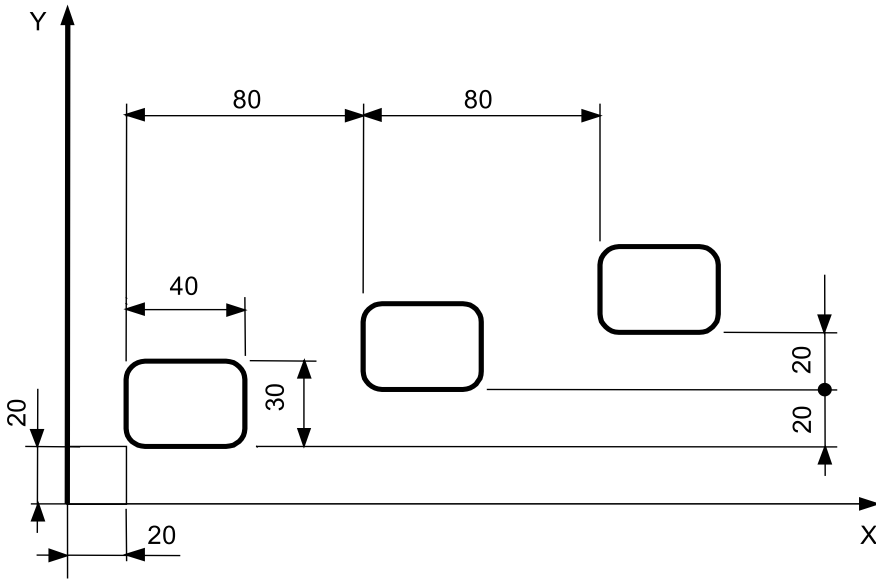
where:

n	Is the number of times the specified block must be executed. It is an integer from 1 to 65535 that can be programmed directly with a number or indirectly with an E parameter. The control allows five nesting levels, i.e. in a repeat block you can program up to four repeat commands.
blocks	It is the set of blocks that must be executed *n* times.

IMPORTANT

The GTO command can be programmed inside an RPT-ERP section. However, if at program end all the cycles programmed with RPT are not executed, the system returns the following error: **'NC063 RPT/ERP CYCLE OPEN AT END OF PROGRAM'**.

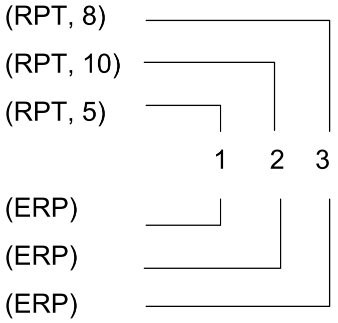
Example 1:
The following is an example of repeat command.



Program:

```
(UGS,X,-50,100,Y,-50,100)
(DIS,"N.3 POCKETS")
(DIS,"MILL D12")
N1  S600 T6.6 M6
N2  (RPT,3)
N3  X40 Y35 M3
N4  Z 2
N5  (RPT,2)
N6  G91 Z-8
N7  G90 G1 G41 X40 Y20 F300
N8  X60
N9  Y50
N10 X20
N11 Y20
N12 G40 X40
N13 Y35 F1000
N14 (ERP)
N15 G Z2
N16 (UIO,X80,Y20)
N17 (ERP)
N18 (UAO,0)
N19 Z20
N20 X Y M30
```

Example 2:
This example shows how three repeat levels are nested.

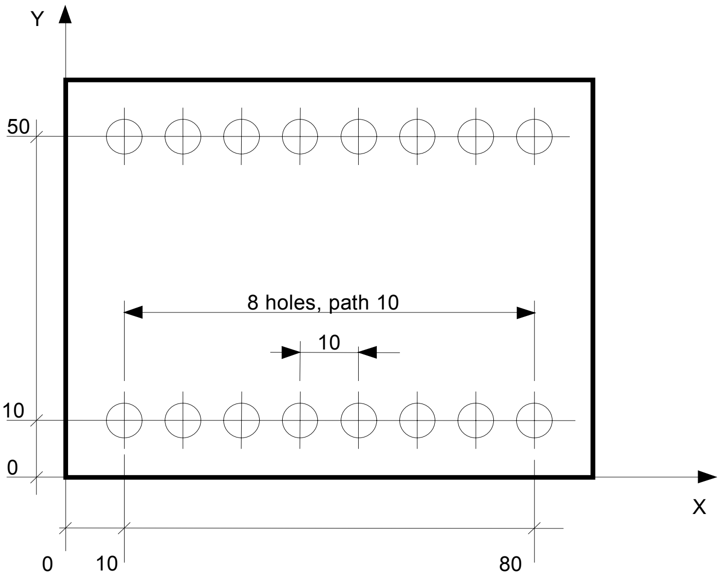


Machining Equidistant Holes

The following example uses a repeat command for machining equidistant holes.

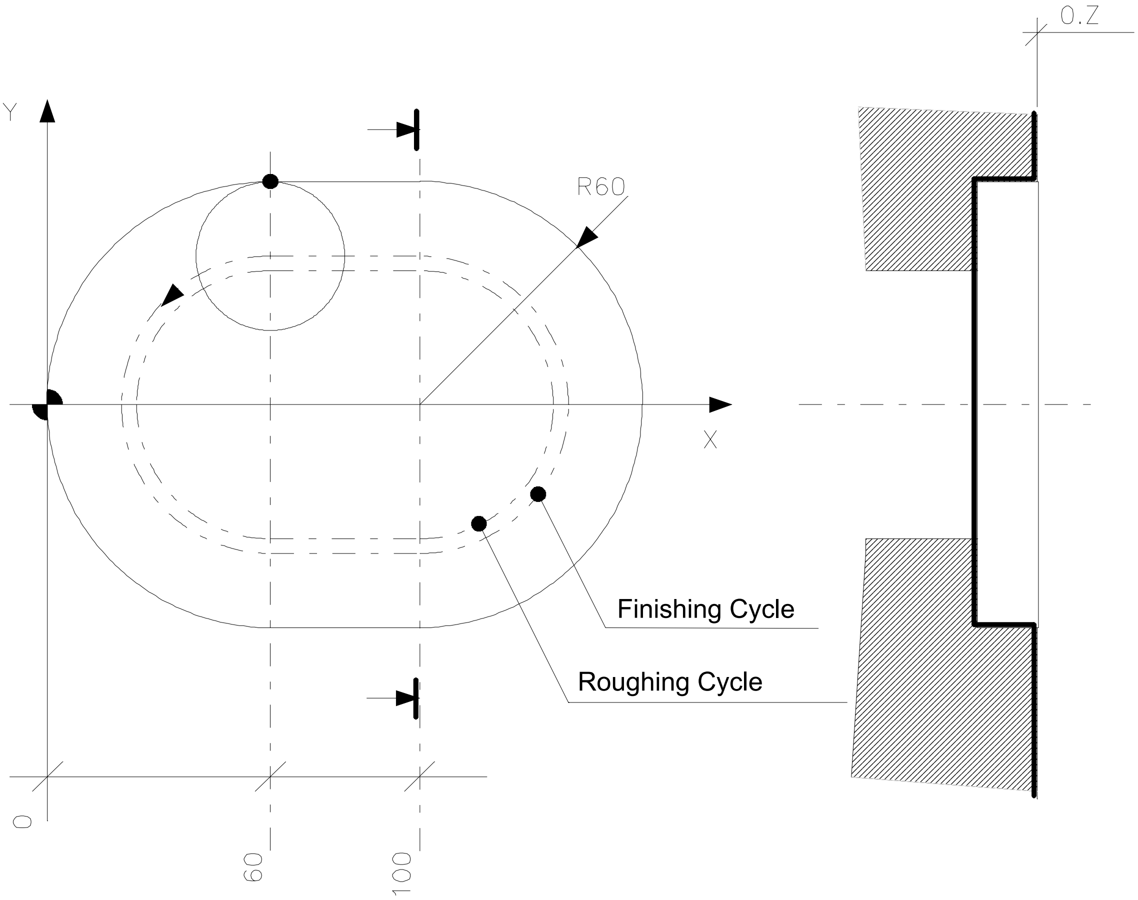
Program:

```
(UGS,X,-50,100,Y,-50,100)
(DIS,"EQUIDISTANT HOLES")
N1  F200 S900 T1.1 M6
N2  G81 R5 Z-10 M3
N3  X10 Y10
N4  (RPT,7)
N5  G91 X10
N6  (ERP)
N7  Y40
N8  (RPT,7)
N9  X-10
N10 (ERP)
N11 G80 G90 XY M5
```



Machining with Roughing and Finishing Cuts

The following example shows repeat commands for machining with one roughing cut and one finishing cut.



Program:

```
(UGS,X,-20,150,Y,-65,60)
(DIS,"DEFINITION OF MACHINING ALLOWANCE")
N1  S350 T6.6 M6
N2  X60 Y M3
N3  Z-50
N4  MSA=0.5
N5  (RPT,2)
N6  G1 G41 X60 Y60 F500
N7  G3 Y-60 I60 J
N8  G1 X100
N9  G3 Y60 I100J
N10 G1 G40 X60
N11 MSA=0
N12 (ERP)
N13 GZ20 M5
N14 X Y M30
```

COMMANDS FOR SUBROUTINE EXECUTION

CLS – CLD - CLT - Call Subroutine

Commands CLS, CLD and CLT recall the execution of a subroutine, i.e. a separate program stored in a file. These commands can call a subroutine from a main program and from another subroutine. Up to four nesting levels are possible.

CLS	Call to a static subroutine:	analysed upon the activation of the main part program
CLD	Call to a dynamic subroutine:	analysed during the execution of the part program and only if the name has not been found earlier
CLT	Call to a temporary subroutine:	analysed during the execution of the part program whenever its call is encountered

Syntax

(CLS,*name***)**
(CLD,*name***)**
(CLT,*name***)**

---

where:

name Subroutine file name.
name must be specified by a string of upper-case characters, or a string variable preceded by the key "?" (question mark).

name may be a 10 SERIES type name (40 alphanumeric characters) or a file with DOS type name (8 characters plus extension and path).

Example:
(CLS,E:\FILE\PROGRAM.PRG) ;call to subroutine with DOS name

Example:
(CLS,PROGRAM.MIO)

NOTE:

- If *name* is a string variable preceded by the "?" code, the subroutine is not analysed upon the activation of the main part program; it is analysed instead during the execution of the latter. This has the following implications, depending on the triliteral used to call the subroutine:
    - CLS: the subroutine cannot contain skip instructions
    - CLD or CLT: the subroutine can contain skip instructions in as much as run times are analysed

- If the MAIN is a 10 SERIES directory (with file names of 48 characters) the subroutine cannot be resident in a DOS directory, and accordingly three-letter code CLS cannot specify a complete *path*.
- The *pathname* to use for calls to subroutines in a DOS directory may also be declared through the three-letter code PTH described later in this chapter. If the pathname is omitted the directory search for the subroutine is carried out as follows :

  1. **No pathname specified via PTH**
  The system searches for the subroutine in the directory in which the calling program is found and if it is not there, it looks in the DOS directories to which logical names have been associated during the characterisation phase of the machine.

  2. **Pathname specified via the PTH instruction**
  The system looks for the subroutine in the directory specified with PTH, if it is not there, it looks in the DOS directories to which logical names have been associated.

- For the **THROUGH mode**: compatibility with the CLS instruction has been maintained, i.e. if the subroutine called contains labels, an error is returned, in as much as this mode does not envisage the activation of the main program upstream. CLD, CLT and paramacros, instead, are managed the same way as the non-through mode.

- The **number of subprograms and "static" labels** managed by the control depends on the corresponding values configured in AMP. For full tables, a memory management procedure is activated, which, to the extent feasible, removes the dynamic and paramacro subroutines from the tables and compacts the static ones (CLS). When it is not possible to free any further space in the memory, a table full message is given out and the system shifts to the same error conditions as we would have during part program selection. At this point, the operator can proceed by either changing the part program or increasing the maximum limits specified for the tables.

- When the tables are full and the **execution of a program is not blocked** thanks to the memory compacting feature, program execution is slowed down, during the subsequent stage of analysis, which inevitably takes place in the event of a blocking error.

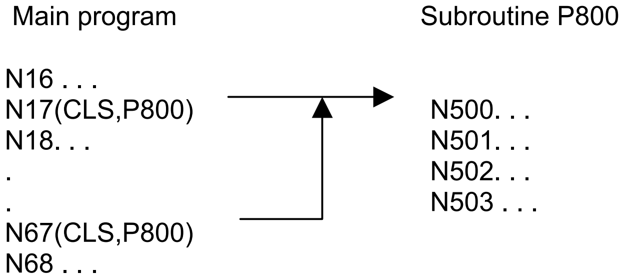
- The successful completion of a program activation stage is no guarantee that the program will be executed without errors, if the memory is full. This is the case, for instance, of a CLD calling a number of CLS instructions large enough to fill up the tables.

- **Re-executing a main program** may generate a tables full message even if the program has been activated in its entirety the first time (after its activation). This is so because after the activation of a program, the tables do not contain the file names and the static subroutine tables called either by static subroutines or by dynamic or temporary subroutines.

- A command for a **call to the same program name** entered through triliterals of different types (e.g. first CLT and then CLS) may generate a label unknown message; such a message can be obviated by replacing the CLS call with a CLD call.

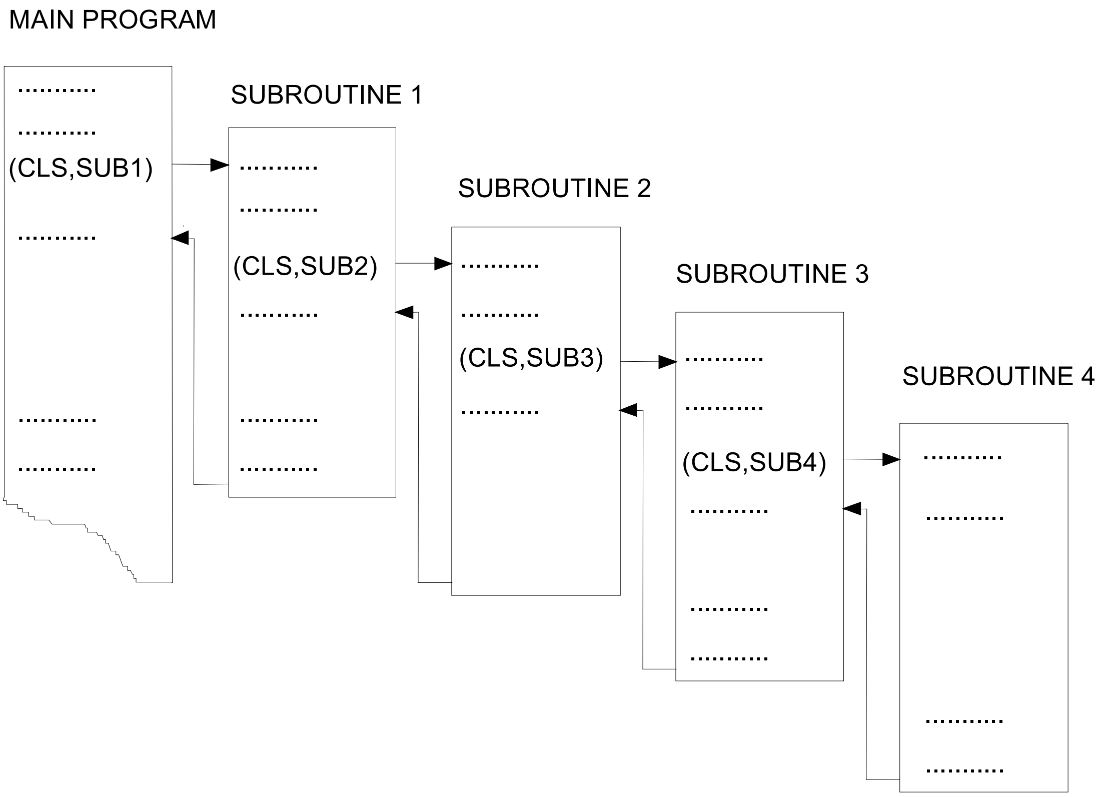
- A **run time change to a subroutine** not followed by the reactivation of the main program will be interpreted only if the call is made through a CLT; otherwise, the one analysed previously will be re-executed, with unpredictable effects on program execution.

Example 1:
The following is an example of a call to a subroutine.



```
Main program                          Subroutine P800

N16 . . .
N17(CLS,P800)                          N500. . .
N18. . .                               N501. . .
.                                      N502. . .
.                                      N503 . . .
N67(CLS,P800)
N68 . . .
```

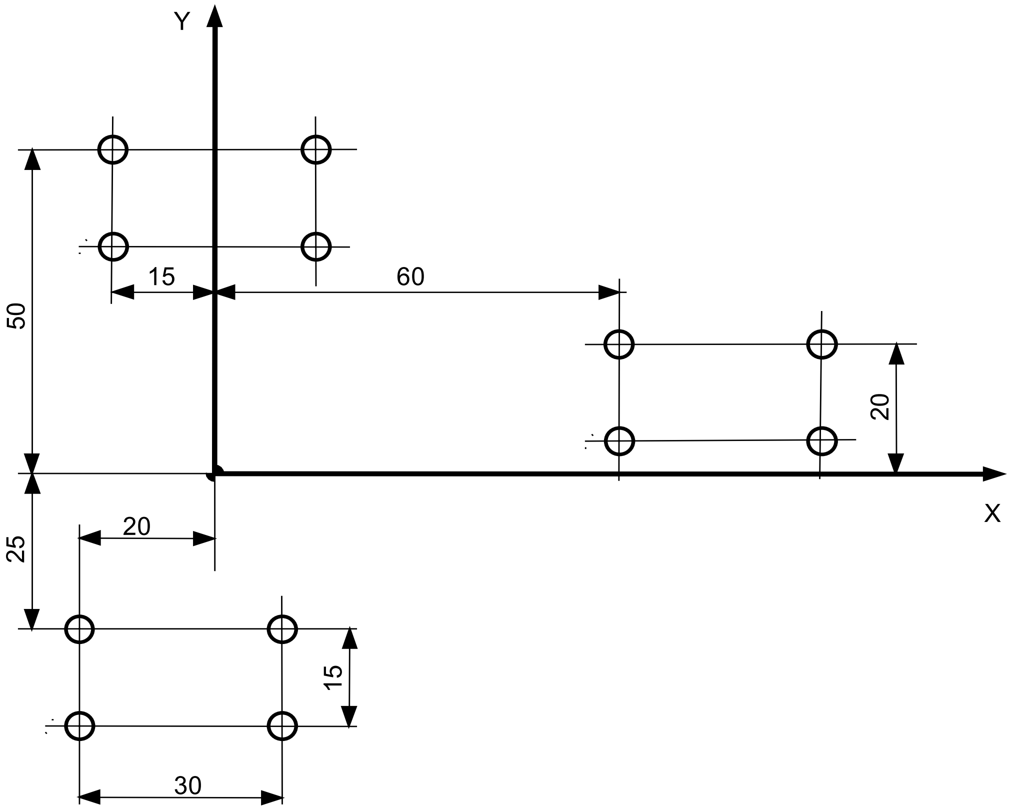
Example 2:
Sequence for execution of four nested subroutines.



Example 3:
Call to subroutine specified in implicit mode

```
SC0.5="PIPPO"
(CLS,?SC0.5)
```

Example 4:
This example uses subroutines for repeated drilling operations.



Main Program:

```
N19 (DIS,"...")
N20 S2000 F180 T2.02M6
N21 (UTO,1,X-20,Y-25)
N22 (CLS,S600)
N23 (UTO,1,X-15,Y50)
N24 (CLS,S600)
N25 (UTO,1,X60,Y20)
N26 (CLS,S600)
N27 Z0
```

Subroutine S600

```
N501 G81 R-108 Z-130 M3
N502 XY
N503 Y-15
N504 X30
N505 Y0
N506 G80
```

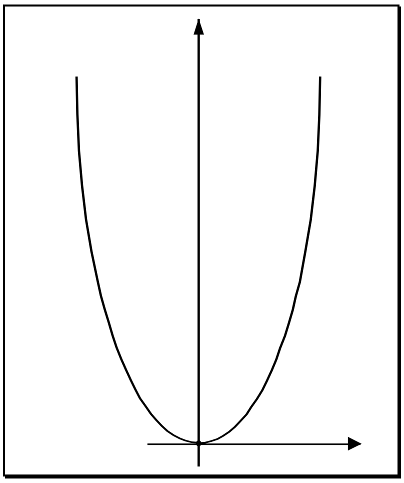
Example 5:
Parabolic profile programming with a parametric subroutine.

Main Program:

```
;N10 MAIN PAR
N20 T1.1M6 S1000 M3 F700
N30 E30=72.795                  ;start X
N40 E31=24.28                   ;focal length (twice the focus)
N50 E32=2                       ;Y increment
N60 E33=108.24                  ;start Y
N70 E34=0                       ;final Y
N80 GX0 Y120
N90 (CLS,PAR)
```

Subprogram PAR:

```
;N500 PAR
;N501 Parametric subroutine: Complete parabola execution, internal profile.
N502 G1 G42 XE30 YE33
N503 E36 = E33
"START" N504
N505 E36=E36-E32
N506 (GTO,END,E36<E34)
N507 E35=SQR( 2*E31*ABS(E36))
N508 XE35 YE36
N509 (GTO,START)
"END" N510
N511 E35=SQR (2*E31*ABS(E34))
N512 XE35 YE34
;N513 Second part of the parabola.
N514 E42=E32
N515 E43=E34
N516 E44=E33
N517 XE35 YE43
"START2" N518
N519 E43=E43+E42
N520 (GTO,END2,E43>E44)
N521 E35= -(SQR (2*E31*ABS(E43)))
N522 XE35 YE43
N523 (GTO,START2)
"END2" N524
N525 E35= - (SQR(2*E31*ABS(E44)))
N526 G40 XE35 YE44
N527 GX0
```



EXAMPLE 6:

Call to the dynamic subroutine, specified in an implicit manner with the PAR subprogram of the previous example

Main program:

```
;N10 MAIN PAR
N20 T1.1M6 S1000 M3 F700
N30 E30=72.795              ;Initial value of X
N40 E31=24.28               ;Focal length
N50 E32=2                   ;Increment of Y
N60 E33=108.24              ;Initial value of Y
N70 E34=0                   ;Final value of Y
N80 GX0 Y120
SC0.5="PAR"
(CLD,?SC0.5)
```

PTH - Declaration of the default *pathname*

The PTH command declares the *pathname* to be used as the default in calls to subroutines and paramacros with DOS names.

Syntax

(PTH,*mode* [,*pathname*]**)**

---

where:

mode	*mode* may have the following values and significance: **mode=0** Taken as the default *pathname* for CLS calls, the *path* of the main program. **mode=1** Declares the *path* to use when there is a (PTH,2) instruction active. This mode may be of use when a *pathname* defined initially with (PTH,1,*pathname*) has to be called numerous times in the part program with (PTH,2). **mode=2** Activates the *path* specified in the instruction or, if none is specified, the one declared previously with a (PTH,1,*pathname*) instruction.
pathname	Path to be declared as the *path* for CLS calls; this is an optional parameter.

NOTE:
The *pathname* declared with three-letter code PTH is preserved even after a RESET of the process in which it was programmed.

EPP - Executing a Portion of a Program

The EPP command allows you to execute a subprogram, i.e. a portion of a program delimited by two blocks with label fields.

Syntax

(EPP,*label1***,***label2***)**

---

where:

label1	Label field of the first block to be executed. A label is a sequence of up to six alphanumeric characters.
label2	Label field of the last subprogram block.

Characteristics:

Program the block label between quotes ("LABEL1") even if in EPP they must be declared without quotes. The system accepts up to 5 nesting levels.

In contouring operations, you can use the EPP command for finish milling with the same program blocks you wrote for roughing. During the roughing phase, use the MSA command to program the machining allowance for finishing.

In positioning operations, you can program points for a centring operation, then use the EPP command to call for different tools in order to execute different operations on each hole. The EPP command can be used to execute a complete machining operation at different orientations on the active interpolation plane.

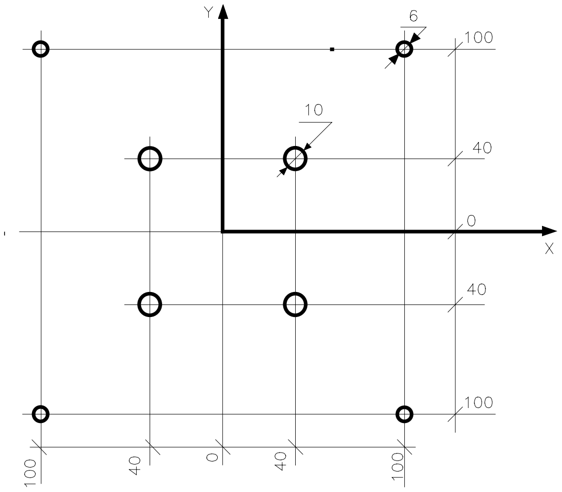
Example 1:

```
   .
   .
"START"N25                 First block with label
   .
   .                       An EPP cannot occur here
   .
"END"N100                  Last block with label
   .
   .
   .
N150 (EPP,START,END)EPP command that specifies the labels. The control executes blocks N25
                           to N100; after this point you must resume execution with the block
                           written after EPP (N150).
```

IMPORTANT

The **'NC062 NESTING OF EPP > 5'** error occurs if more than five instructions are nested in an EPP command.
The GTO command can be programmed inside an EPP section. However, if at program end all the cycles programmed with RPT are not executed, the system returns the following error: **'NC063 RPT/EPP CYCLE OPEN AT END OF PROGRAM'**.

Example 2:
This example shows how to use the EPP command in a positioning operation:



Program:

```
(UGS,X,-110,110,Y,-110,110)
N1 (DIS,"DRILLING CENTRE HOLES")
N2 F300 S2000 T1.1 M3 M6
N3 G81 R0 Z-3
"D6"N4
N5 X100 Y100
N6 X-100
N7 Y-100
N8 X100
"D10"N9
N10 X40 Y40
N11 X-40
N12 Y-40
N13 X40
"END"N14
N15 G80
N16 (DIS,"BIT D6")
N17 F200 S1800 T2.02 M3 M6
N18 G81 R Z-22
N19 (EPP,D6,D10)
N20 G80
N21 (DIS,"BIT D10")
N22 F220 S1600 T3.3 M3 M6
N23 G81 R Z-24
N24 (EPP,D10,END)
N25 G80
```

EPB - Execute Part-Program Block

The three-letter code EPB allows execution of a part-program block.
Execution can be conditioned by the result of a comparison specified within the command; If the condition is not met an alternate part-program block may be defined so as it can be executed.

Syntax

(**EPB**, *part_program_block1 [, par1 operator par2 [, part_program_block2])*

---

where:

part_program_block1	Is the part-program block to be executed. This can be a string in inverted commas or a character, local or system variable.
par1	Is a local or system variable or a constant whose value is compared to the value contained in the *par2 parameter.*
operator	Logic operators which can be used in the expressions : = equal to < less than > greater than <> not equal to <= less than or equal to >= greater than or equal to
par2	Is a local or system variable or a constant whose value is compared to the value in the parameter *par1.*
part_program_block2	Is the block of the part-program to execute only if the specified condition for *part_program_block1* is not met. It is an optional parameter. This can be a string in inverted commas or a character, local or system variable.

Characteristics:

If *par1 ,operator* and *par2* conditions are not specified, the program executes the *part_program_block1 block.*

Only one level of nesting is accepted by the system with the three-letter code EPB.

Example:
(EPB, “(EPB,’E1=1’)”) : accepted by the system
(EPB, “(EPB,’(EPB,SC0.100)’)”) : NOT accepted by the system (error: FORMAT ERROR)

The part-program blocks specified in the three-letter code EPB are not analyzed in the activation phase of the part-program, therefore it is up to the programmer to see that these blocks do not cause any malfunctioning of the part-program.

Example:
(EPB,”(CLS,SUBROUT)”) : the subprogram SUBROUT is not pre-analyzed in the activation phase, therefore it cannot contain any jump (GTO) instructions.

(EPB,’ “LABEL” ’) : The label LABEL will not be inserted in the label table of the part-program.

(EPB, “ (DIS, E1)”) : equivalent to the block (DIS, E1)

SC0.30= ” E1 =10”
(EPB, SC0.30) : equivalent to the block E1=10

SC40.30 = “# X10”
(EPB, SC40.30) : equivalent to the block #X10

(EPB,”(CLS,SUBROUT)”,E1=34) : calls the subprogram SUBROUT only if E1=34

(EPB,” E1=100”, SN1=25,” E1=0”) : assigns 100 to E1 only if SN1=25, otherwise assigns 0 to E1

(EPB,”(EPP,LAB1,LAB2)“,SC0.2=“OK”,“(EPP,LAB3,LAB4)“) : executes from the label LAB1 to the label LAB2 if SC0.2 is equal to OK, otherwise executes from the label LAB3 to the label LAB4.

(EPB, “(EPB,’ E1 = 2’, E0 < 100)”, E0 > 70) : assigns 2 to E1 only if E0 is between 70 and 100.

(EPB,“ (EPB,’ E0 = 5’, E0 < 5) ",E0 < 10,E0=10) : assigns the value 5 to E0, if E0<5; assigns 10 to E0 if E0>10.
If 5<E0<0, E0 does not change.

BRANCHING AND DELAY COMMANDS

GTO - Branch Command

The GTO command performs a jump to the specified block if the condition expressed in the command is true.

Syntax

(GTO,*label* [**,***par1 operator par2*]**)**

where:

label	Label of the program block to branch to. A label is a string of up to 6 alphanumeric characters. In the destination block the label must be between quotes while in GTO command the label must be programmed without quotes. Also, the label can be a character variable preceded by "?"
par1	It is the local or system variable whose value is compared to the value of the *par2* variable.
operator	Logical operators that can be used in expressions: = equal < less than > greater than <> different than <= less than or equal to >= greater than or equal to
par2	It is the local or system variable whose value is compared to the value of the *par1* variable.

Characteristics:

If *par1, operator* and *par2* are not specified, the program always jumps to the block marked by the label.

The label the part program must branch to can also be expressed as a local or system variable (preceded by the "?" character).

This facilitates execution of programs having a great number of blocks. In particular, by programming the label as a variable it is possible to program a default label and call it "DEFLAB": The part program positions on "DEFLAB" if the label in the variable does not exist.

Example 1:

```
;SC0.3 contains the positioning label

(GTO,?SC0.3)
"ONE"
.
.
.
.
"TWO"
.
.
.
.
"THREE"
.
.
.
"DEFLAB"
.
.
.
```

If SC0.3 does not contain "ONE", "TWO" OR "THREE" the part program positions on the "DEFLAB" block. If the "DEFLAB" label exists, the **'NC054 Undefined label'** error is displayed and part program execution is interrupted.

Example 2:

N01 (GTO,START)	the part program always branches to "START"
N10 (GTO,END,E1 > 123)	branch to "END" if the value of E1 is greater than 123
N20 (GTO,LAB1,@COOLANT = 1)	branch to "LAB1" if the PLUS variable @COOLANT is on
N30 (GTO,START,E1 <> E5)	branch to "START" if the value of E1 is different from that of E5
N40 (GTO,LAB1,SC1.2H = "OK")	branch to "LAB1" if the two characters beginning with SC1 are equal to OK
N50 SC1.3="ABC"	prepares the variable for the following block
N60 (GTO,?SC1.3)	branch to ABC label

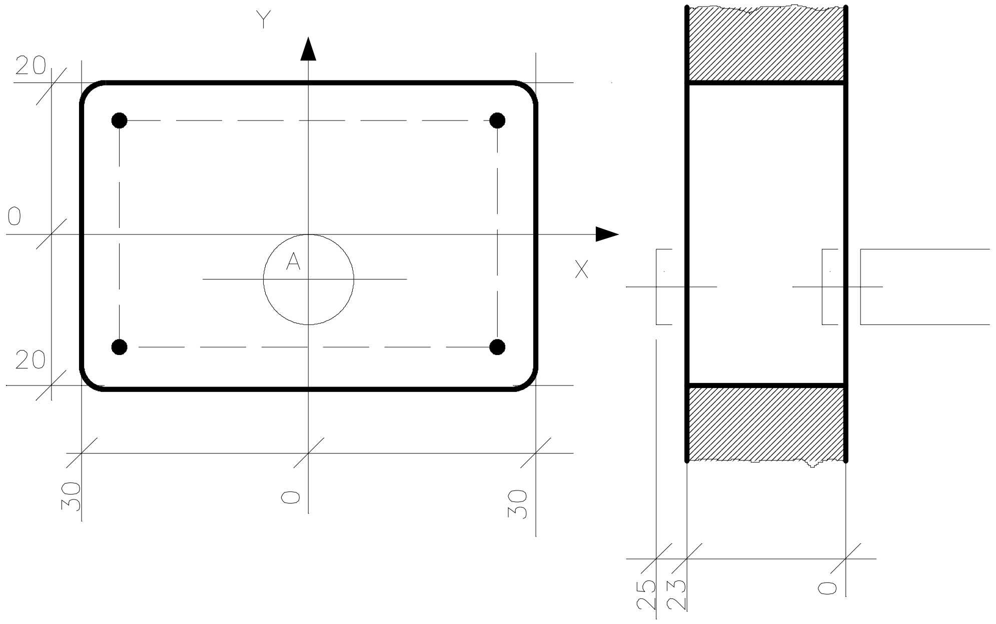
Example 3:
The instruction:

(GTO,END,SC2.3H = "ABC")

branches to the "END" label in the program if the three characters (.3) beginning from SC2 are ABC.

NOTE:
The character string to be compared in the branch command must always be programmed between quotes.

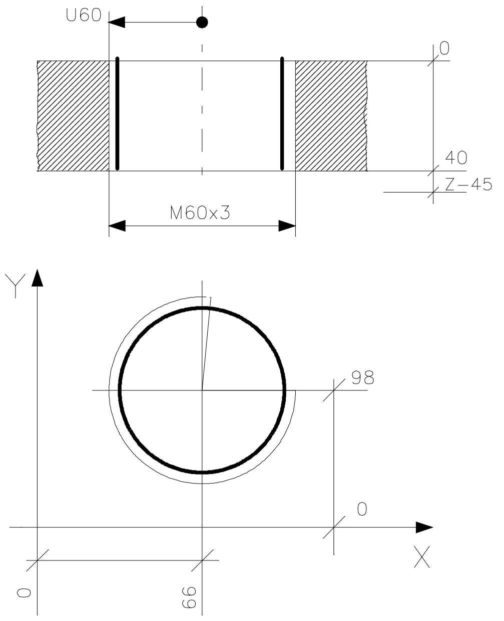
Example 4:
Here is an example of conditional branching in slot milling.



Program:

```
(UGS,X,-50,50,Y,-50,50)
N1 (DIS,"MILL A SLOT")
N2 F500 S2000 T1.1 M3 M6
N3 E31=-3.5
N4 E32=-24
"START"
N6 G X Y-10
N7 Z(E31)
N8 G1 G42 X Y-20
N9 X-30
N10 Y20
N11 X30
N12 Y-20
N13 G40 X
N14 Y-10
"END"
N16 E31=E31-3.5
N17 (GTO,START,E31>E32)
N18 E31=-25
N19 (EPP,START,END)
N20 G Z10
```

Example 5:
This is an example conditional branching in cylindrical thread machining.



E30 = Diameter of Next Cut
E31 = Cut Depth Increment (Diameter)
E32 = Return Diameter
E33 = Final Diameter

Program:

```
N1 (DIS,"THREAD DIA 60")
N2 S150 T5.5 M6
N3 G0 X66 Y98 Z5 M3
N4 E30=56.8
N5 E31=0.5
N6 E32=50
N7 E33=60
"I" N8
N9 G0 Z5
N10 U(E30)
N11 G33 Z-45 K3
N12 GU(E32)
N13 E30=E30+E31
N14 (GTO,F,E30>E33)
N15 (GTO,I)
"F" N16
N17 GU(E32)
N18 Z5
N19 U(E33)
N20 G33 Z-45 K3
N21 GU(E32)
N22 Z20
```

IF ELSE ENDIF

The IF command allows the opening of a part-program session which will be executed only if a condition specified within the command is met. This part-program session must terminate with the ENDIF command : the ELSE command can be optionally included in the IF command, which delimits, up until the ENDIF command, a part-program session which will be executed if the condition specified in the IF command is not met.

Syntax

(IF, *par1 operator par2* **)**
(ELSE)
(ENDIF)

where:

par1	Is a local or system variable or a constant whose value will be compared with the *par2* parameter.
operator	Logic operators which can be used in the expressions :

=	equal to
<	less than
>	greater than
<>	not equal to
<=	less than or equal to
>=	greater than or equal to

par2	Is a local or system variable or a constant whose value is compared to the value in the parameter *par1.*

Characteristics.

If the condition specified in the IF command is met, the blocks specified up until the ELSE command (if present) or until the ENDIF command (if the ELSE command is not present) will be executed, otherwise they will be ignored.

The blocks between the ELSE command and the ENDIF command will be executed only if the condition specified in the IF command is not met.

32 levels of nesting are allowed with the IF ELSE ENDIF command.

Every IF command must have a corresponding ENDIF command.

Examples:

```
( INP,” E0 VALUE “ ,30 , E0)
( IF , E0 = 3)
( DIS , “ E0 is equal to 3 “)
(ELSE)
( IF E0 > 3)
( DIS , “ E0 is greater than  3 “)
(ELSE)
( DIS , “ E0 is less than  3 “)
( ENDIF)
( ENDIF)
```

```
( INP,” E0 VALUE“ ,30 , E0)
( IF , E0 >10 )
( IF , E0 >20 )
( IF , E0 >30 )
( IF , E0 >40 )
( IF , E0 >50 )
( DIS , “ E0 is greater than 50  “)
( ENDIF)
( ENDIF)
( ENDIF)
( ENDIF)
( ENDIF)
```

DLY - Defining Delay Time

The DLY command specifies a delay in program execution.

Syntax

(DLY,*time***)**

where:

time	Delay time in seconds (minimum value 0.1). You can program the delay with a numerical constant or an E parameter.

DSB - Disable Slashed Blocks

This command enables/disables slashed blocks.

Slashed blocks have "/" symbol as first character and their execution is conditioned by the DBS value.

Syntax

DSB= *value*

---

where:

value	The value may be:	
	DSB=0	slashed blocks are executed
	DSB=1	slashed blocks are not executed

REL - Releasing the part program

The REL command releases the part program and all subroutines.

Syntax

(REL)

Characteristics:

The REL function disables the part program. It can be entered through MDI.

After the program has been released, the following message will be displayed: ***'NC058 END OF PROGRAM'***.

WOS - WAIT on signal

Syntax

(WOS,*par1 operator par2***)**

---

where:

par1	Is a local or system variable or a constant to be compared to the value of *par2*.
operator	Logic operators that can be used in expressions:
	= equal to
	< lower than
	> higher than
	< > different than
	<= lower or equal to
	>= higher or equal to
par2	Is a local or system variable or a constant to be compared to the value of *par1*.

Characteristics:

The process where the WOS function is programmed goes on WAIT until the condition is satisfied.

By default the WOS command is synchronised.

If the system is on WAIT during program debugging (because a WOS function has been programmed), it is possible to assign to the *par1* variable the value that satisfies the condition specified in WOS.

To make this assignment, follow these steps:

- Press CYCLE STOP
- Key in the correct value of the *par1* variable.
- Press CYCLE STOP

Example:

```
(WOS,E1>4)
(WOS,SC10.5 = "GOOFY")
(WOS, TOOL_STATUS>=E3)
(WOS,!USER_VAR3.5CH 0 SC4.5)
```

DEVICE DEFINING COMMANDS

GDV - Definition of the *device* for file access

This three-letter code is used to reserve use of drive A for file read/write operations and to define remote *devices* on which to work.

Syntax

(GDV, *device***)**

where:

device Name of *device*.
May be a letter or, if preceded by a "?" key, a local or system variable specifying the drive to be used.
For the floppy drive, the letter "A" must be used as the identifier.
For remote devices, the name to be given is that with which it was defined with the option MINI DNC E65/E66 (e.g. "K").

Characteristics:

If drive A is already being used by another process or UTILITY (e.g. DOS SHELL), the error **NC251 Driver busy or not configured** is generated; this error can be handled by the part program (for further details, refer to appendix C).

Examples:
(GDV,A)

Reserves use of drive A.

SC0.1='A'
(GDV,?SC0.1)

Use of drive A is reserved, as specified in variable SC0.1

IMPORTANT A system RESET cancels access to drive A but not to remote devices.

RDV - Release *device*

This three-letter code is used to release drive A or the remote *device* defined previously with the three-letter code GDV, so as to make it available for other users.

Syntax

(RDV, *device***)**

---

where:

device	Name of *device*. May be a letter or, if preceded by a ? key, a local or system variable specifying the drive to be used. For the floppy drive, the letter "A" must be used as the identifier. For remote devices, the name to be given is that with which it was defined with the option MINI DNC E65/E66 (e.g. "K").

Characteristics:

A process reset is tantamount to automatic execution of a (RDV,A) command.

Examples:

```
(RDV,A)
```

Releases drive A previously allocated with GDV.

```
SC0.1='A'
(RDV,?SC0.1)
```

Similar to the previous example, but with the drive specified in variable SC0.1.

END OF CHAPTER

Chapter **13**

HIGH SPEED MACHINING

GENERAL CONSIDERATIONS

The High Speed Machining feature is used for machining surfaces (profiles) defined by points, created by CAD/CAM systems, on machine tools having 3 to 5 axes (3 linear + 2 rotary).

The High Speed Machining feature must be enabled in the AMP environment, by selecting the appropriate field in the PROCESS CONFIG section (**PROC CHAR** softkey).

WARNING

This feature may only be enabled for the first 4 processes.

To program this feature, proceed as follows:

1. Create a setup file (part program) that contains all the parameters for handling the profile: tools, axes and kinematics of the machine.

   The configuration file will be recalled in the main program through a three-letter command, with the following format:

*(**HSM**, setup file name)*

2. Create the profile by inserting it directly in the main program:

   Example:

```
; HSM PROGRAMMING EXAMPLE
G0 X..Y..Z.. A.. B.. F…
------
(HSM, CONFIG1)
------
(CLS,PROFILO1 ------------------------------------------→  G61
------                                                     G1 X…Y..Z..A..B..
------                                                     -------------------------
; END OF PROGRAM                                           -------------------------
------                                                     G60
```

The profile may also be inserted in a specific file which will be recalled as a subprogram by the ***CLS*** instruction.

Example 2:

```
; HSM PROGRAMMING EXAMPLE
G1 X..Y..Z.. A.. B.. F…
 -----
(HSM, CONFIG1)
G61
G1 X..Y..Z..A..B..
 -----
G60
; END OF PROGRAM
```

PROGRAMMING POINTS AND CHARACTERISTICS OF THE PROFILE

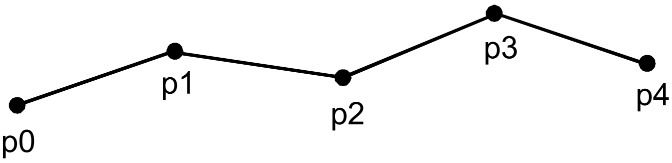
A profile is a set of points that make up the ISO part-program of the surface to be machined, created by the CAD/CAM system in which the characteristics described in this chapter are to be respected.

On the basis of the programmed points a polynomial curve will be constructed, defining the path to be followed. This path will pass through the programmed points with a configurable tolerance. The methods by which the points are to be linked will be defined by the G01 and G00 codes that may be programmed together with the points.

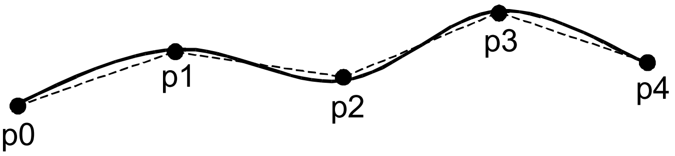
IMPORTANT

To ensure that the polynomial curves are calculated correctly, we recommend the points be programmed with at least 5 figures after the decimal point (e.g. 10.37854); programming with fewer figures may result irregularities on the profile.

The sections executed in **G00** will be considered as individual positioning operations; each point will be linked to the next by means of a “linear” movement to be performed with the dynamic traverse positioning (each section in G00 will start at zero speed and end at zero speed). For this reason, G00 mode will not calculate the polynomial curve. At the end of a section in G00 there will be no pause (end of movement synchronism, entry into tolerance status, etc..) and the next movement will be carried out immediately. This behaviour is similar to the programming of G01 and G09 codes in the same block.



With **G01,** each point will be linked geometrically to the following ones by means of a polynomial curve, so the generated path may be considered “**continuous**”. This link will be interrupted by the programming of a G00 or the programming of special G codes described below. The dynamics of the sections in G01 are the same as the “cutting” movements (such as normal G01 movements in ISO programming).



In addition to the **G01** and **G00** functions, the following G functions, specific for the HSM feature, may be programmed in the profile.

G61
Determines the start of the profile and must be programmed in a block on its own. When the G61 function is activated, there must be no form of virtualization active (UPR,UVP,UVC,TCP).

Before activating the G61 function, the setup file must be defined by means of the instruction:

*(**HSM**, setup file name)*

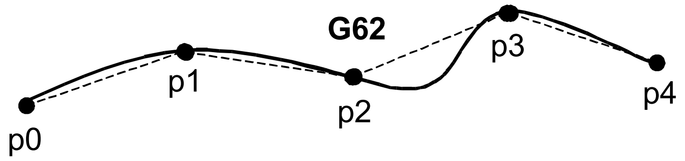
The G61 command may ONLY be executed within a part program in AUTO or BLK BY BLK status.

G60
Determines the end of the profile and must be programmed in a block on its own.

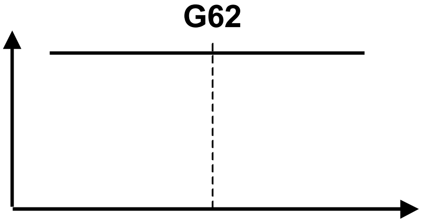
If the machine is in single STEP execution, the profile between G61 and G60 is considered as a single instruction. To stop its execution, it is necessary to switch to HOLD status.

G62
Splits a profile in two parts and determines the point where one profile ends and another begins, maintaining continuity between the two curves.

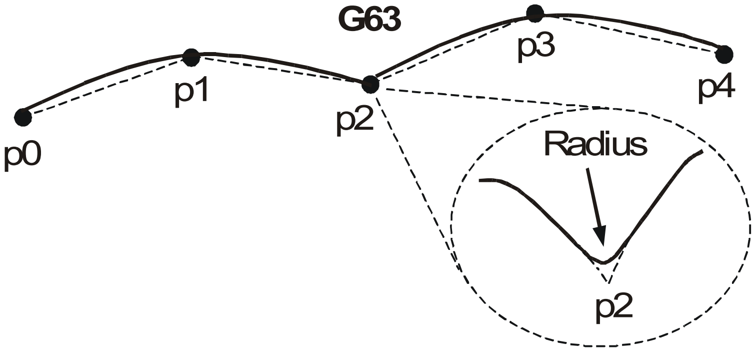
The points preceding the G62 function will be used to generate a first curve, while the subsequent points will be used to generate another one. These curves **will be** linked and will therefore be continuous; the initial inclination of the second curve will correspond to the final inclination of the previous curve.



As regards dynamics, with G62 no deceleration and acceleration ramp will be generated to link the two curves. This G function must be programmed in a part program block on its own.

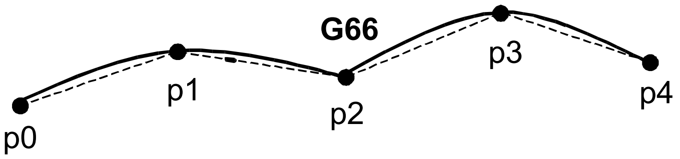


G63
Splits a profile into two parts and determines the point where one profile ends and the other one begins, **maintaining continuity** between the two curves. The points preceding the G63 function will be used to generate a first curve, while the subsequent points will be used to generate another one. While with G62, the initial inclination of the second curve depends strictly on the final inclination of the first, with G63, the initial inclination of the second curve **IS NOT** influenced by that of the first. To maintain continuity, a "radius" that depends on the chordal error with which the splines are to be calculated is inserted.

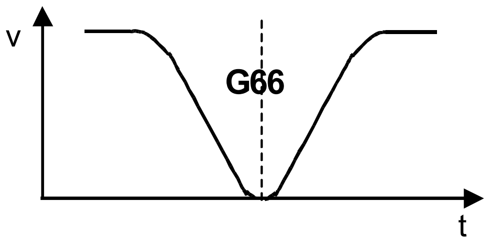


With G63 a reduction in speed may occur at the point where the two curves are linked. This G function must be programmed in a part program block on its own.

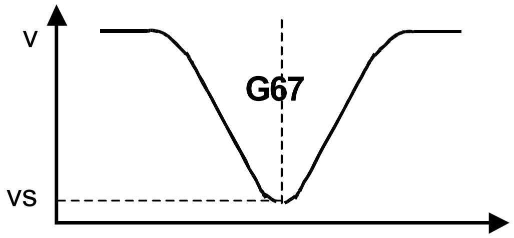
G66
Splits a profile into two parts and determines the point where one profile ends and the other one begins, creating a **discontinuity** between the two curves, that is, the point preceding the G66 represents an edge. At this point, two curves are generated, the first using the points preceding the G66 function and the second using the subsequent points. These curves **<u>will NOT be</u>** linked, and so there will be a discontinuity.



This discontinuity will be reached at zero speed; the first curve will therefore end with a deceleration ramp to 0 (zero) speed after which the second curve will be tackled with an acceleration ramp to reach the required machining speed. This G function must be programmed in a part program block on its own.



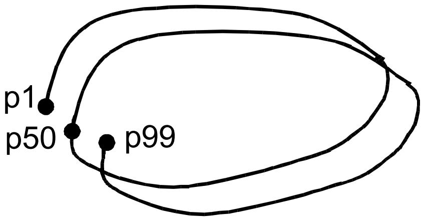
G67
With **G67**, a "discontinuity" may be defined on the profile defined with G66. What changes is the dynamic approach to the edge, that is, the end of the curve is not reached at zero speed but at a speed value (vs) that enables the axes to reach the edge without any dynamic problems. This speed value is calculated on the basis of the acceleration that may be withstood by each axis. This G function must be programmed in a part program block on its own.



Considerations on the use of the G62,G63,G66 and G67 functions (transition codes)

The transition G codes are particularly useful when "similar", repetitive curves are to be defined (providing the programmed points are also similar and repetitive).

Supposing we have a profile defined by 100 points of which the first 50 represent the first machining pass (from **p1** to **p50**) and the other 50 (from **p50** to **p99**) the same profile shifted slightly (second pass).



As the points between **p1** and **p50** are "similar" to the points between **p50** and **p99**, the conditions for calculating the two polynomial curves will also be similar. Two "parallel", almost identical curves will therefore be generated.

If the G62 function has not been programmed on point p50 the NC may generate curves that are not perfectly parallel. This normally undesired effect is due to the fact that the calculation of the polynomial takes into account the "history" along the calculated paths.

The "history" of point p1 is clearly different from that of point p50. In fact, point p1 has no history while in point p50, the NC has followed a path determined by the first 49 points.

When the G62 function is inserted, it cancels the "history" and produces a geometrical pattern almost identical to the one calculated starting from point p1.

In machining processes that entail several passes, failure to program G62 would have the undesirable effect of producing different levels of machining between one pass and another.

GENERAL HIGH SPEED MACHINING PROGRAMMING STRUCTURE

Between the G61 and G60 blocks it will only be possible to program the points that make up the profile to be machined or the G codes for defining their management method: no other type of programming will be allowed.

Points may be programmed using, absolute programming may be used by means of **(G90)** or incremental programming by means of **(G91)**. All numerical parameters required may be defined directly or by means of **E** or **L** variables: programming with expressions is not valid so XE(E2) or X(E1+E2) type programming is not allowed, while XE1 is allowed.

The syntax of the allowed program lines will be:

N… [**G00** | **G01**] [**G90** | **G91**] [points] F….

N… [**G62** | **G63** | **G66** |**G67**]

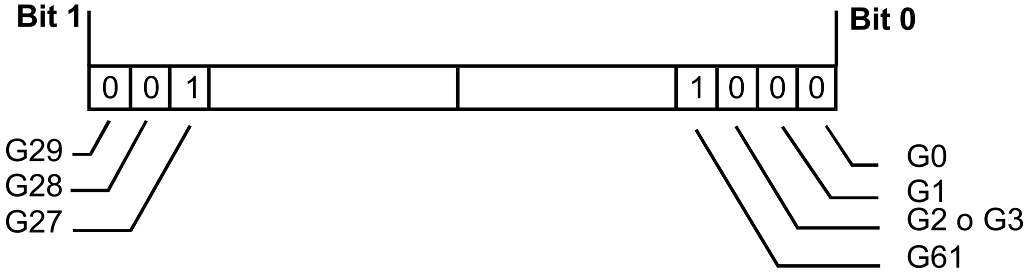
Activation of the HSM (High Speed Machining) feature **G61** forces of G01 and G90, modes while at the exit (**G60**) the G functions active when G61 was programmed will be restored.

The first point programmed **MUST** be expressed in absolute positions (G90) and must contain the programming of all axes associated with the HSM programming (axes configured in the HSM setup file).

Interaction with Machine Logic

The G61/G60 program section will be considered, from the system point of view, as a single program block. As regards interfacing with the machine logic, a request for consent for movement will be made when the G61 function is reached, and an end of movement request will be made when the G60 function is reached (in the same way as for the G27 and G28 continuous movements).

A regards consent for movement, the **XW03** variable, which contains the type of movement, will be set as shown below:



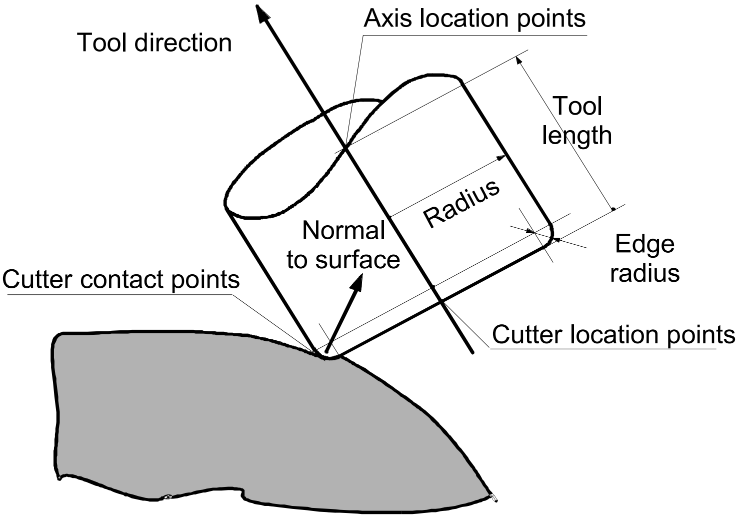
POINT DEFINING CONVENTIONS

Points and machining coordinates

Before defining how the points are handled, it is necessary to specify what they represent as programming may be executed in relation to three types of coordinates, that is:

- **Cutter Contact Points**, which refer to the actual cutting point
- **Cutter Location Points**, which refer to the point normally indicated as the centre of the tool
- **Axis Location Points**, which refer to an arbitrary point fixed to the machining axes

The cutter contact points are linked to the cutter location points through the geometry and orientation of the tool. The axis location points are linked to the cutter location points through the geometry of the machine tool. In machining processes with three axes, the coordinates will simply be translated while, in those with five coordinated axes, rototranslation matrices that take into account the geometrical transformations due to the movement of the rotary axes will be applied. The figure below shows what is meant by cutter contact points, cutter location points and axis location points.



Points are defined by means of normal axis coordinates in the format [**Axis name**][**Position**]; example: X100 Y200 Z40.

Tool Direction: IJK

The tool direction represents the orientation of the tool (from the tip to the attachment) within the part reference system.

WARNING

In the sections that follow, versor means a vector of unit length.

Two methods may be used to define the tool direction. The first is by directly programming the versor that identifies the tool direction. This versor is expressed using the **ijk** coordinates in the format:

[**i**] [**X-coordinate component**] [**j**][**Y-coordinate component**] [**k**][**Z-coordinate component**]

The system will automatically normalize the length of the versor to the unitary length (1.0).

The second way of defining the tool direction is by programming the rotary axes. The system will automatically determine the three components of the **ijk** versor depending on the kinematics of the machine.

Normal to the Surface Direction: MNO

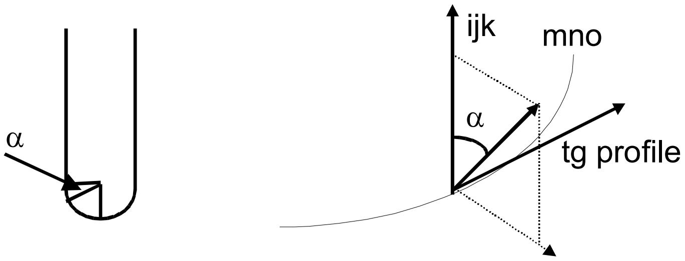
The normal to the surface direction represents the direction of the “line” perpendicular to the surface to be machined (starting from the surface) within the part reference system.

There are two ways of defining the direction normal to the part. The first is by directly programming the versor that identifies the normal direction. This versor (of a unitary length) is expressed using the **mno** coordinates in the format:

[**m**] [**X-coordinate component**] [**n**][**Y-coordinate component**] [**o**][**Z-coordinate component**]

The system will automatically normalize the length of the versor to the unitary length (1.0).

The second possibility consists of asking the system to determine this direction automatically. The direction will be calculated on the basis of the tangent to the profile (displacement direction), on the basis of tool direction (vector **ijk**) and angle of contact between the part and the tool (angle α). This angle is conventionally assumed to be 0° if the tool touches the part with its tip, 90° if the tool touches the part with its left side and -90° if it touches it with its right side (in the example below, α=90°). Thanks to this calculation, the **mno** versor is normal to the tangent to the profile and defines an angle α (angle of contact) with the tool versor **ijk**.



This type of approach is only significant when the contour is to be machined. When a surface is to be machined, this approach could fail as there is no information about the “surface” to be machined, only information about the “direction” of displacement.

Tool Radius Application Direction: PQD

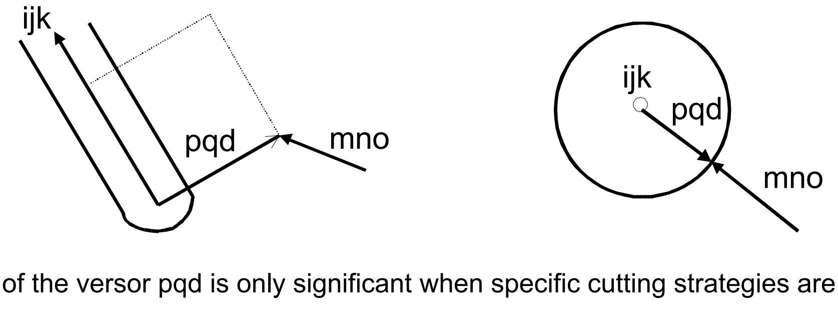
The direction of application of the tool radius represents the direction in which radius compensation is to be applied (starting from the centre of the tool) within the part reference system.

There are two ways of defining the tool radius application direction. The first is by directly programming the versor that identifies the direction. This versor (of a unitary length) is expressed using the **pqd** coordinates in the format

[p] [X-coordinate component] [q][Y-coordinate component] [d][Z-coordinate component]

The system will automatically normalize the length of the versor to the unitary length (1.0).

The second way is to have this direction calculated automatically by the system. The direction is calculated automatically on the basis of the tool direction (**ijk** versor) and the normal to the part (**mno** versor). This calculation ensures that the **pqd** versor is normal to the tool direction and is on the plane formed by the **ijk** and **mno** versors.



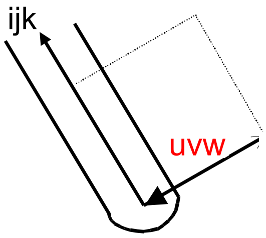
Programming of the versor pqd is only significant when specific cutting strategies are applied.

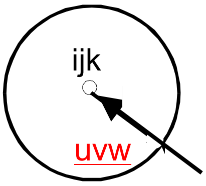
Paraxial Compensation Factors: UVW

These factors can be used instead of mno and pqd versors for determining tool radius compensation: instead of working perpendicularly to the programmed profile (mno), this compensation is obtained by moving the programmed point along the uvw direction by a quantity equivalent to the radius. The module of the uvw vector need not be unitary.
In order to define the paraxial compensation factors two techniques can be adopted. One consists of programming directly the vector that identifies this direction, as expressed through the **uvw** coordinates in this format:

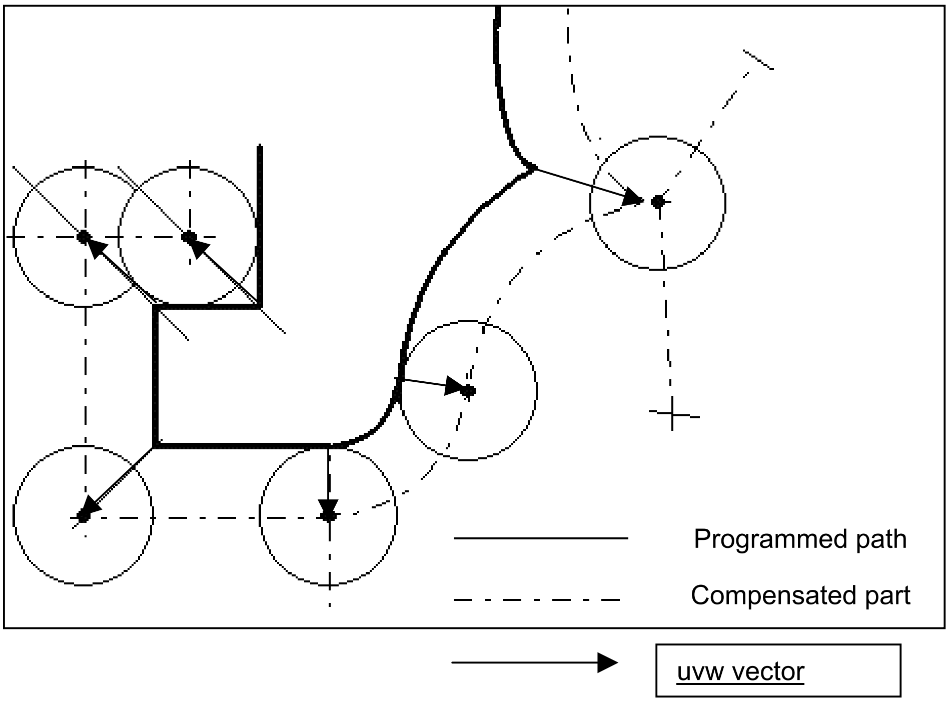
[u] [Abscissa Component] [v][Ordinate Component] [w][Vertical Component]

The other technique consists of having the system calculate the direction automatically. The direction is calculated through a similar procedure as is used for the mno versor, but angle α can only assume the following values: +90°, if the tool touches the work piece with the left side; –90° if the tool touches the work piece with the right side. For this reason, the management of uvw vectors is compatible with cylindrical tools as well as with spherical routers working on the side. Toroidal tools are not admitted. Furthermore, the uvw vector differs from mno at corners where the length of the vector is multiplied by an appropriate correction factor.



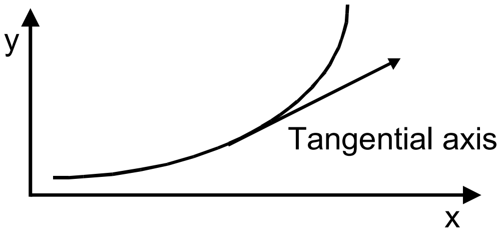


The basic difference with respect to mno versors, in fact, is observed at the "pointed" parts of the profile, as shown in the figure below.



Tangential Axis

The tangential axis is an axis whose position is calculated so as to remain tangential to the profile described. It is calculated on the basis of the tangent to the polynomial curve on the work plane.



An initial value of the tangential axis (first programmed point) may be defined and the subsequent positions may be calculated on the basis of this value.

FEATURES PROVIDED BY HIGH SPEED MACHINING

Depending on the type of machine tool used, the points programmed and a series of additional parameters, different features may be obtained using “High Speed Machining”.

Generic machines or machines with 3 axes are characterized by the fact that they do not have axes for orienting the tool during the machining phase. The tool direction is generally fixed.

Tool Radius and Length Compensation

In order to use tool radius and length compensation, the type of axis positions and vectors as defined in the table below must be included in the part program. The feed rate may refer to the point of contact between the tool and the part (chip removal speed, therefore at the Cutter Contact Point) or the centre of the tool (at the Cutter Location Point). The difference between these two types of settings is significant when tools with a large diameter are used, that is, where the path of the centre of the tool is significantly different from that of the profile of the part.

Setting	Description
Axis positions	The points must be expressed in **Cutter Contact Points**
Tool Direction	The tool direction vector is defined during the set-up phase (on a 3-axis machine, it is fixed and defined by the TOD parameter) and remains unchanged throughout the machining process. ijk programming is ignored.
Normal to Surface	The mno vector must be programmed or automatically calculated by the system
Radius Application	The pqd vector must be programmed or automatically calculated by the system

Example:

```
N001 G61
N002 G1 X10Y10Z10  m0n0o1 p0q0d1 F10000
N002 X20Y10Z10 ….
…..
N100 G60
```

Tool Length Compensation

In order to use tool length compensation, the type of axis position and vectors as defined in the table below must be included in the part program. The feed rate will refer to the centre of the tool (at the Cutter Location Point). However, starting from the points referred to the tool centre, it is possible to recalculate the Cutter Contact Points to be able to apply the tool diameter compensation (see TOL configuration parameters) and to define the parameters of feed rate relative to the profile actually machined (on the Cutter Contact Point).

Setting	Description
Axis Position	The points must be expressed as **Cutter Location Points**
Tool Direction	The tool direction vector is defined in the set-up phase (on a 3-axis machine, it is fixed and defined by the TOD parameter) and remains unchanged throughout the machining process. ijk programming is ignored.
Normal to Surface	It is necessary to program the mno vector when you want to apply tool diameter compensation.
Radius Application	It is necessary to program the pqd vector (or request its automatic determination) when you want to apply tool diameter compensation.

Example:

```
N001 G61
N002 G1 X10Y10Z10 F10000
N002 X20Y10Z10 ….
…..
N100 G60
```

No Tool Compensation

In this case, only the positions of the axes as defined in the table below have to be included in the part program. The feed rate will refer to the actual movement of the axes (Axis Location Point). However, starting from points referred to the part dimensions, it is possible to recalculate the tool centre points (Cutter Location Point) to apply tool length compensation and, if necessary, recalculate the Cutter Contact Points in order to apply tool diameter compensation (see the TOL configuration parameters). Thus, it will be possible to set feed rate relative to the motion of the tool centre (on the Cutter Location Point for tool length compensation) or relative to the machine profile (on the Cutter Contact Point).

Setting	Description
Axis Positions	The points must be expressed in **Axis Location Points**
Tool Direction	The tool direction vector is defined during the set-up phase (on a 3-axis machine it is fixed and defined by the TOD parameter) and remains unchanged throughout the machining process. It is **only** used if the setting of the feed rate in relation to the centre of the tool has been requested, otherwise it is ignored. The setting of ijk is ignored.
Normal Surface	It is necessary to program the mno vector when you want to apply tool diameter compensation.
Radius Application	It is necessary to program the pqd vector (or request its automatic determination) when you want to apply tool diameter compensation.

Example:

```
N001 G61
N002 G1 X10Y10Z10 F10000
N002 X20Y10Z10 ….
…..
N100 G60
```

Tangential Axis Management

When management of the tangential axis is requested, its position is calculated automatically by the system. Any programming of the tangential axis within the part program defines further rotations of the axis with respect to the tangent calculated by the system. The programming of the value 0 (zero) on the axis activates the calculation of the position by the High Speed algorithms.

SETUP

The configuration of the HSM environment is contained in a specific setup file: all dimensional parameters must be expressed in the unit of measurement of the machine tool.
This setup is activated whenever the G61 code is programmed.
All the numerical values of the part program can be defined directly or through **E** or **L** parameters.

The setup file may be divided into four sections:

- a section of *General three-letter codes*
- a section of *Axis Setup three-letter codes*
- a section of *Machine Configuration three-letter codes* or machine kinematics definition section.
- a section of *PathOptimizer Configuration three-letter codes*

General Three-Letter Codes

(**PNT**,Type,Param,Poly)	Definition of types of points
(**VER**,Ijk,Mno,Pqd,uvw)	Versor management
(**JRK**,Enable,Jrs,Entities,Tstab,Tintgr,Ttop)	Look Ahead management
(**THR**,Tol,TolV,Scale,NullMov,Chord, ErrRot,MaxLen)	Threshold programming
(**TOL**,Len,Radius,EdgeRad,Angle,OriginLen,Rorig, RgOrig)	Tool definition
(**TOD**,Xcomp,Ycomp,Zcomp)	Tool direction (3D)
(**CRV**,Len,Ratio,Mode)	Change in curvature management
(**EDG**,Angle,VAngle,Mode,Acc)	Edge management

Axis Setup Three-Letter Codes

(**AXI**,Name,Id,Type,CinType)	Definition of types of axis
(**AXP**,Name,NullMov,Pitch,Lim-,Lim+)	Definition of axis parameters
(**DIN**,Name,Vmax,Amax,Jmax,Vrap,Arap,Jrap)	Definition of axis dynamics

Machine Configuration three-letter codes

(**MAC**,Type,Bed,Um)	Defines machine type
(**CIN**,Name,Xoff,Yoff,Zoff,Xrot,Yrot,Zrot)	Defines machine kinematics

PathOptimizer Configuration three-letter codes

(**SMT**,Toll,TollX,TollV,Racc,RaccX,RaccV,C0,C0X,C0V)	Smoothing parameters and tolerances
(**SMS**,Split,SplitX,SplitV,Cond,CondX,CondV,Len,Pnt,C1,Progr,Rcnc)	Special smoothing parameters

Type of points described in the part program

The syntax of the three-letter code that defines the type of points described in the part program is as follows:

(PNT, Type, Param, Poly, Format) *example: (PNT, CLP,CLP,QUI, PNT)*

Parameter	Type	Values	Description
Type	Character	CCP CLP AXI Obligatory	Defines the type of points described in the Part Program **CCP** Cutter Contact Points: the points entered are defined in “Cutter contact points”, so tool compensation (radius and length) may be executed on these points. **CLP** Cutter Location Points: the points entered are defined in “Cutter location points“, so tool compensation (length only) may be executed on these points. **AXI** Axis Location Points: the points entered are defined in “Axis location points”, so no type of compensation may be executed on these points.
Param	Character	CCP CLP AXI Obligatory	Defines the type of profile for which the feed rate is programmed in the Part Program and depends on the type of points entered. The feed rate always refers to the “3D” profile. Any rotary or additional axes present are not involved in the calculation of the feed rate but “follow” the execution of the 3D profile. The feed rate is however limited when the latter axes, in following the 3D axes, tend to exceed their dynamic limits. **CCP** The programmed Feed rate refers to the profile generated on the “Cutter contact points”; this setting is clearly only valid if the points entered are of the CCP type. **CLP** The programmed Feed rate refers to the profile generated on the “Cutter location points”; it may be used for the CCP and CLP type points, while for AXI type points it is only valid when the tool direction can be determined (either from the programming of the rotary axes or from the programming of the ijk versors). **AXI** The programmed Feed rate refers to the profile generated on the “Axis location points”; it may only be used for AXI type points.
Poly	Car.	CUB or QUI Obligatory	Defines the degree of the polynomial curve generated for the execution of the programmed profile. **CUB** A cubic polynomial is generated. **QUI** A quintic polynomial is generated.
Format	Characters	PNT, POL or BSP selectable	Defines the programming method used for a profile, i.e., the input format selected. **PNT** Points **POL** Polynomials **BSP** B-Spline

Versor management methods

If programmed, the versors must refer to the abscissa, ordinate and vertical axes, respectively. The syntax of the three-letter code defining the management modes for these versors is as follows:

(VER, Ijk, Mno, Pqd, Uvw) *example: (VER , REL , PRG , PRG)*

(VER , ROT, , REL , PRG)

Parameter	Type	Values	Description
Ijk	Character	PRG or REL Obligatory	Defines whether the “Tool Direction” vector is programmed directly by means of the ijk components or whether it is to be calculated on the basis of the position of the rotary axes. When the versor is not necessary, this setting is ignored. **PRG** The versor is programmed using the ijk components. **REL** The versor is to be determined from the programmed position of the rotary axes. **ROT** The versor is to be determined from the current position of the rotary axes.
Mno	Character	PRG or REL Optional	Defines whether the “Normal to Surface” vector is programmed directly using the mno components or whether it is to be calculated automatically by the system. If the versor is not necessary, this setting is ignored. **PRG** The versor is programmed using the mno components. **REL** The versor is to be determined automatically by the system. If omitted, this means that a uvw vector is used.
Pqd	Character	PRG or REL Obligatory	Defines whether the “Tool Radius Application Direction” vector is programmed directly using the pqd components or whether it is to be calculated automatically by the system. When the versor is not necessary, this setting is ignored. **PRG** The versor is programmed using the pqd components. **REL** The versor is to be determined automatically by the system.

Uvw	Character	PRG or REL Optional	Indicates whether the “Tool Radius Compensation Factor” has been programmed directly through the uvw components or has to be calculated by the system automatically. When the versor is not required, the programming mode is disregarded. **PRG** The vector is programmed through the uvw components. **REL** The vector has to be calculated automatically by the system. If omitted, this means that an mno versor is used.

NOTE:
if tool radius compensation is required, vectors MNO and UVW cannot both be omitted.

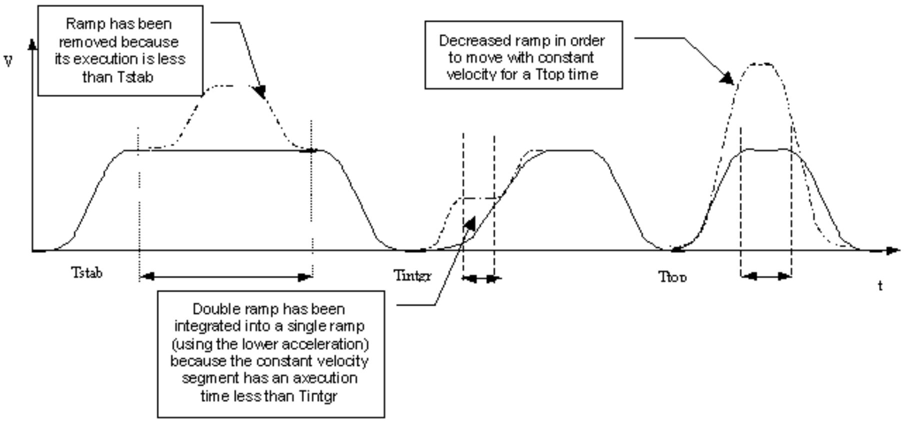
Look Ahead management

The syntax of the three-letter code that defines the method of look ahead and dynamics management is as follows:

(JRK, Enable, Jrs, Entities, Tstab, Tintgr,Ttop) *example: (JRK, ENA,,)*

Parameter	Type	Values	Description
Enable	Character	ENA,DIS or AXI Obligatory	Defines the type of dynamics to be used. **ENA** Enables the use of Jerk Limitation (Mov=8). The specified jerk refers to the dynamics described on the profile. **DIS** Disables Jerk Limitation, only uses “S ramps” (Mov=2). **AXI** Enables the use of Jerk Limitation (Mov=8). The specified jerk refers to the dynamics described by the axes and how this affects the dynamics of the profile. The use of this method is closely associated with the quality of the points entered (number of decimals and distribution).
Jrs	Number	Taken from JRS variable Optional	Modifies the value of the JRS variable of the system only for the G61 section being processed; if the value is omitted, the JRS variable active in the system is used.
Entities	Number	Optional	The High Speed algorithm uses a dynamic queue of 64 elements (polynomial curves) and starts movement after calculating the 64th. The start may be adjusted by programming the number of elements to be calculated before starting movement. 0 means maximum number of elements.
Tstab	Number	Optional	Defines the time window (in ms) within which the speed smoothing algorithm is to be applied. This algorithm removes unnecessary accelerations so as to avoid machine oscillations. (See Figure 1)

Parameter	Type	Values	Description
Tintgr	Number	Optional	Defines the maximum time (in ms) for integrating the acceleration (or deceleration) phases with the sections at a constant feed rate. (See Figure 1)
Ttop	Number	Optional	Defines the minimum time (in ms) for executing a section with a constant feed rate V. (See Figure 1)



Ramp has been removed because its execution is less than Tstab

Decreased ramp in order to move with constant velocity for a Ttop time

Double ramp has been integrated into a single ramp (using the lower acceleration) because the constant velocity segment has an execution time less than Tintgr

Picture 1

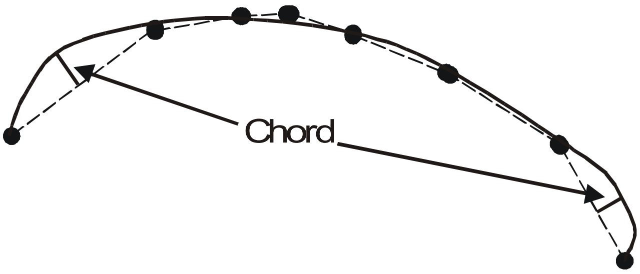
Thresholds

The syntax of the three-letter code that defines the value of thresholds used in generating the polynomial curves is as follows:

(THR, Tol, TolV, Scale, NullMov, Chord, ErrRot, MaxLen)
example: (THR , 0.01 , 0.0001 , 0 , 0 , 0.1)

Parameter	Type	Values	Description
Tol	Number	Obligatory	Defines the tolerance range to be used in generating the polynomial curves. As defined previously, the polynomial curves generated pass through the programmed points, within a given tolerance threshold. This parameter defines the tolerance value. It is defined in mm (or inches if the machine is configured in inches).
TolV	Number	Obligatory	Defines the tolerance range to be used in generating the polynomial curves calculated on the versors. This value is important because the precision with which the rotary axes are positioned depends on the precision with which the curves are generated on the versors (in particular, on the ijk versor). The number has no dimension in that it refers to entities like versors which always have a dimension of 1. We suggest a value equal to 0.1 times the Toll value defined previously be set. 0 disables the management of a tolerance on the versors.
Scale	Number	Obligatory	Scale factor to be applied to the programmed axes. If 0 is set, the scale factor is not to be applied. It is not applied to the versors and rotary axes, when programmed.
NullMov	Number	Obligatory	Null movement threshold. If the distance between one point and the next is less than this threshold in mm (or inches if the machine is configured in inches) the next point is eliminated.
Chord	Number	Obligatory	Maximum allowed chordal error at input and output during the generation of a polynomial. Value expressed in mm (or inches if the machine is configured in inches).
ErrRot	Number	Optional	Maximum admissible chordal error for the determination of polynomials on rotary axes. It is expressed in degrees.

Parameter	Type	Values	Description
MaxLen	Number	Optional	Maximum length of generated polynomials. Value expressed in mm (or inches). It allows a major control of the position tolerance. (Toll) Advised values >5 mm, lower values could slow the working.



Tool definition

In the High Speed Machining system, “Cylindrical”, “Ball-ended” and “Toroidal” type tools may be managed. The syntax of the three-letter code that defines the characteristics of the tool to be used for machining is as follows:

(TOL, Len, Radius, EdgeRad, Angle, OriginLen, Rorig, RgOrig) *example: (TOL , , , 1 , -90 , 0)*

Parameter	Type	Values	Description
Len	Number	Taken from the offset active on G61 Optional	Defines the length of the tool to be used for tool length compensation. If no value is set, the tool length is taken from the offsets active in the system on the activation of the G61. The value set here is only active during the machining of the current section of G61/G60.
Radius	Number	Taken from the offset active on G61 Optional	Defines the radius of the tool to be used for tool radius compensation. If no value is set, the tool radius is taken from the offsets active in the system on the activation of G61. The value set here is only active during the machining of the current section of G61/G60.
EdgeRad	Number	Obligatory	Defines the radius on the edge of the tool. As this value is not handled by the system offsets, it has to be set.
Angle	Number	Optional	Defines the angle of contact between the tool and the part. It is used in the automatic calculation of the vector normal to the surface.
OriginLen	Number	Optional	Defines the length of the tool for which the part program was generated. This field is used when the points entered refer to “Axis location points” and you wish to apply the compensation for tool length.
ROrig	Number	Optional	Defines the radius of the tool for which the part program was generated. This field is used when the points entered refer to “Axis location points” and you wish to apply the compensation for tool length and diameter.
RgOrig	Number	Optional	Defines the radius of the tool for which the part program was generated. This field is used when the points entered refer to “Axis location points” and you wish to apply the compensation for tool length and diameter.

Tool direction (3D)

Defines the tool direction (to be used for compensation purposes) for generic machines or machines with 3 axes (it is not required on machines with 5 axes). In practice, the unitary vector (similar to ijk) which identifies the tool direction in the part reference system has to be defined.
The syntax of the three-letter code that defines the direction of the tool to be used for machining is as follows:

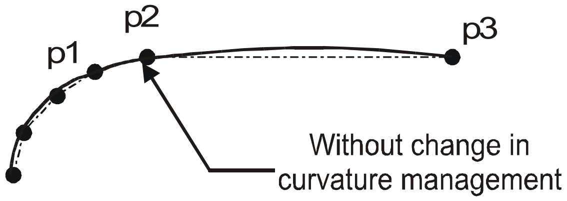
(TOD, Xcomp, Ycomp, Zcomp) *example: (TOD , 0 , 0 , 1)*

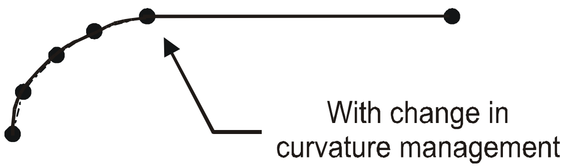
Parameter	Type	Values	Description
Xcomp	Number	Obligatory	Component of the tool direction along the X axis.
Ycomp	Number	Obligatory	Component of the tool direction along the Y axis.
Zcomp	Number	Obligatory	Component of the tool direction along the Z axis.

Change in curvature management

The algorithm used by the CAM systems to generate the points of a profile takes into account the “chordal error”, that is, it generates points closer together when the radius of curvature of the profile to be described becomes more accentuated. A change in curvature may therefore be generically identified by a variation in the distance between one set of points and the following ones. This variation in curvature may be identified automatically by the High Speed Machining algorithms so as to avoid oscillations around the point where this change takes place.

Supposing, for example, we have a curved section followed by a straight section. The spline on the straight section would tend to oscillate or generate a “hump”, so it is important to identify it.





At the "curvature change" point, the system will automatically introduce a G62 or a G63 depending on user requests. As an alternative, it is possible to add more points (closer spacing) in the longer portion; the number of points added depends on the ratio between the lengths of the two portions (long/short).

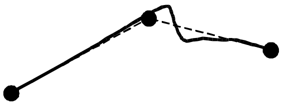
The syntax of the three-letter code that defines how to identify and then manage the change in curvature is as follows:

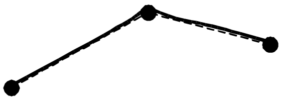
(CRV, Len, Ratio, Mode) *example: (CRV , 1 , 6 , G63)*

Parameter	Type	Values	Description
Len	Number	Obligatory	Defines the minimum length of the “long section” for the change in curvature to be managed. It is defined in mm (or inches if the machine is configured in inches).
Ratio	Number	Obligatory	Defines the ratio between the long section and short section so that the change in curvature may be identified. For example, the value 6 defines that the distance between p2 and p3 must be greater than 6 times the distance between p1 and p2 for the change in curvature to be activated.
Mode	Character	G62 or G63 or PNT Obligatory	Defines the transition code to be set on the “change in curvature” point **G62** The two segments are generated with two splines tangential to one another, so the second spline will have an initial inclination equal to the final inclination of the first spline. **G63** The two segments are generated by two non-tangential splines, but are linked on the basis of the calculation tolerance set. **PNT** Additional points (closely spaced) are added on the longer portion.

Edge management

The automatic identification of edges is important for the same reason as the identification of changes in curvature. Failure to identify edges could generated incorrect oscillations on the splines. The optimum dynamic approach for handling an edge is to stop at zero speed and then restart on the next section. Stopping may be damaging however as the tool, as it turns, continues to remove material and so some "notches" may be visible on the part. For this reason, it is possible to define whether and how to stop at the edge.

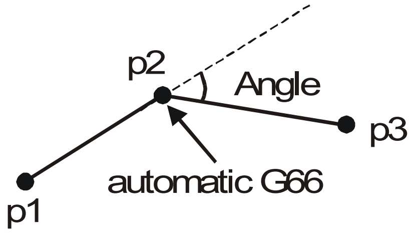


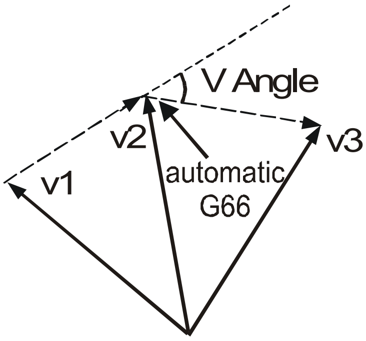


The syntax of the three-letter code that defines how to identify and therefore how to manage the presence of edges is as follows:

(EDG, Angle, VAngle, Mode, Acc) *example: (EDG , 30 , 0 , G66 , 1)*

Parameter	Type	Values	Description
Angle	Number	Obligatory	The system is capable of automatically determining the edges defined by the programmed points (edges on the linear axes) and automatically programming a G66. This value, expressed in degrees, defines the threshold angle beyond which a point is defined an "edge" point. See the figure below.
VAngle	Number	Obligatory	It is similar to the previous value and is to be used for the versors. The presence of an edge on the versors generates an "edge" in the movement of the rotary axes. It is therefore advisable to add a G66 also in these cases. See the figure below.
Mode	Character	G63, G66 or G67 Obligatory	Defines the transition code to be set on the edge **G63** The edge is executed by inserting a link which is generated by taking the configured chordal error into account. **G66** Movement ends at zero speed. **G67** Movement ends at the maximum speed at which the edge may be faced in the best possible way.
Acc	Number	Obligatory	Acceleration that may be withstood by the axes in tackling the edge: 1 means that an axis may withstand an increase in speed equal to 1 acceleration. It only applies to G67.





Axis definition

The axes to be subjected to the High Speed Machining algorithms may be defined. A maximum of 6 axes may be defined, of which 3 axes make up the three Cartesian axes, 2 are rotary axes (for machines with 5 axes) and other additional axes. The syntax of the three-letter code for defining the axes is as follows:

(AXI, Name, Id, Type , CinType, Diam, LinRot) *example: (AXI , X , 1 , ABS , LI1, DIS)*

example: (AXI , C, 4, OTH, OTH, LIN)

Parameter	Type	Values	Description
Name	Character	Obligatory	Defines the name of the axis. This name may be different from the one set in AMP for the axis, and may therefore be compared to an implicit "DIN" in the HSM setup. This association only applies to the G61/G60 section being machined.
Id	Number	Obligatory	Defines the ID of the axis. This axis must be under the control of the Process in which the part program is being executed.
Type	Character	ABS, ORD, VRT, OTH TAN or TAO Obligatory	Defines the axis type, and influences the method by which the axis is managed by the HSM algorithms. **ABS** The axis is of the x-coordinate type (in the Cartesian coordinate system) **ORD** The axis is of the y-coordinate type (in the Cartesian coordinate system) **VRT** The axis is of the z-coordinate type (in the Cartesian coordinate system) **TAN** Tangent axis, calculated automatically by the system **TAO** Tangent axis, calculated automatically by the system, with the axis moving (positive direction) clockwise, and hence in a direction opposite to that of the trigonometric convention **OTH** Other type of axis, different from the previous ones.

CinType	Character	LI1, LI2, LI3, WRK, TOL or OTH Obligatory	Defines the position of the axis in the kinematic chain of the machine (see following sections). For generic machines or machines with 3 axes for which the kinematic chain does not have to be defined, we recommend the following setup: **LI1** To be associated with the axis defined as ABS (x-coordinate) in the previous field. **LI2** To be associated with the axis defined as ORD (y-coordinate) in the previous field. **LI3** To be associated with the axis defined as VRT (z-coordinate) in the previous field. **OTH** Additional axis. For machines with 5 axes, the setup values are as follows: **LI1** First axis in the kinematic chain. **LI2** Second axis in the kinematic chain. **LI3** Third axis in the kinematic chain. **WRK** Rotary axis closest to the part. **TOL** Rotary axis closest to the tool. **OTH** Additional axis.
Diam	Character	ENA or DIS Optional	Makes it possible to characterise the axis as Diametral. If this is not defined, the corresponding parameter currently active in the system is used. **ENA** Axis defined as diametral. **DIS** Normal axis.
LinRot	Character	LIN or ROT Optional on CN Required on Path Optimizer	Makes it possible to specify the type of added axis, as either linear or rotary. If no value is set, for an execution on the CN, the corresponding parameter currently active in the system is used; for an execution on the Path Optimizer, an error is notified. **LIN** Linear type added axis. **ROT** Rotary type added axis.

Axis parameters

Some characteristics of the axis set on the system may be varied. These variations are only active in the G61/G60 section being machined. The syntax of the three-letter code is as follows:

(AXP, Name, NullMov, Pitch, Lim-, Lim+) *example: (AXP , X , 0 , , ,)*

Parameter	Type	Values	Description
Name	Character	Obligatory	Axis name as defined in the three-letter code (AXI).
NullMov	Number	Obligatory	Defines the null movement for the axis in question. If the position programmed for an axis differs from the current position (or last programmed position) for a value lower than the null movement, the movement of the axis (the new position) is not considered. We recommend it be left set to 0 and activated only in cases in which the part program under execution contains imprecisions in the programming of the axes.
Pitch	Number	Taken from the system Optional	Used for redefining the rollover pitch for the axis in question. It may be used, for example, to program a non-rollover rotary axis with rollover values and vice versa.
Lim-	Number	Taken from the system Optional	Used for redefining the lower operating limit for the axis in question.
Lim+	Number	Taken from the system Optional	Used for redefining the upper operating limit for the axis in question.

Axis dynamics

Some dynamic characteristics of the axis set in the system may be varied. These variations are only active in the G61/G60 section being machined. The syntax of the three-letter code is as follows:

(DIN,Name, Vmax, Amax, Jmax, Vrap, Arap, Jrap) *example: (DIN , X , , , , , ,)*

Parameter	Type	Values	Description
Name	Character	Obligatory	Axis name as defined in the three-letter code (AXI).
Vmax	Number	Taken from the system Optional	Used for redefining the maximum speed at which the axis may be moved when a machining operation in G01 is in progress. It must be defined in mm/min.
Amax	Number	Taken from the system Optional	Used for redefining the maximum acceleration at which the axis may be moved when a machining operation in G01 is in progress. It must be defined in mm/sec^2.
Jmax	Number	Taken from the system Optional	Used for redefining the maximum Jerk to which the axis may be moved when a machining operation in G01 is in progress. It must be defined in mm/sec^3.
Vrap	Number	Taken from the system Optional	Used for redefining the maximum speed at which the axis may be moved when a machining operation in G00 is in progress. It must be defined in mm/min.
Arap	Number	Taken from the system Optional	Used for redefining the maximum acceleration at which the axis may be moved when a machining operation in G00 is in progress. It must be defined in mm/sec^2.
Jrap	Number	Taken from the system Optional	Used for redefining the maximum Jerk at which the axis may be moved when a machining operation in G00 is in progress. It must be defined in mm/sec^3.

Example

Setup for machine with 3 axes:

```
(PNT , AXI , AXI , QUI )
(VER , REL , PRG , PRG )
(JRK, ENA ,  ,  ,400 ,150 ,200)
(THR , 0.01 , 0.0001 , 0 , 0.001 , 0.05 )
(TOL ,  ,  , 0 , -90 ,  )
(TOD , 0 , 0 , 1 )
(CRV , 1 , 6 , G63 )
(EDG , 30 , 0 , G66 , 1 )
;
(AXI , X , 1 , ABS , LI1 )
(AXP, X , 0 ,  ,  ,  )
(DIN , X ,  ,  ,  ,  ,  ,  )
;
;
(AXI ,Y, 2 , ORD, LI2 )
(AXP,Y, 0 ,  ,  ,  )
(DIN ,Y,  ,  ,  ,  ,  ,  )
;
;
(AXI , Z , 3 , VRT, LI3 )
(AXP, Z , 0 ,  ,  ,  )
(DIN , Z ,  ,  ,  ,  ,  ,  )
;
```

END OF CHAPTER

Chapter 14

MULTIPROCESS MANAGEMENT COMMANDS

GENERAL

This chapter discusses interprocess commands, i.e. instructions that allow the programmer to synchronise parallel part program execution various processes or to shift the "axes" resource between processes in order to meet specific requirements.

The following table summarises these commands:

3-LETTERS CODE	FUNCTION
DCC	Defines remote channels
PVS	Defines the remote PLUS variables environment
PRO	Specifies the default process to which synchronisation codes and commands must be sent
SND	Sends a synchronisation message to the specified process
WAI	Puts the current process on wait until another message arrives from another process
EXE	Activates automatic part program execution by the specified process
ECM	Executes an MDI block in the specified process
GTA	Shifts the "axes" resource among processes

IMPORTANT By default SND and WAI are synchronised.

SYNCHRONIZATION AMONG PROCESSES

Notes On The "Wait" Function:

The synchronisation messages received by a process are recorded in a memory area called MESSAGE QUEUE. This local area is process-dedicated: there is one message queue for each process.

If the (WAI,Pn) instruction is executed by a part program EXE, the process must wait for a message from process <n>. The system looks for the message in the MESSAGE QUEUE:

- ***If the message is not found in the MESSAGE QUEUE,*** the process puts itself on WAIT and suspends part program execution.
- ***If the message is in the MESSAGE QUEUE***, part program execution continues and the message is deleted from the message queue.
- ***If the queue contains a message coming from a process other than <n>,*** the message is not deleted but it is kept aside in order to be utilised when a WAI for the appropriate process is programmed.
  The process can exit from WAIT mode only when the queue contains a message coming from the process specified in WAI.

Notes On The "Send" Function:

Three-letter code SND allows synchronisation messages to be sent to the specified process; the messages may be synchronous or asynchronous:

If the message is synchronous, the process that sent SND goes on WAIT and waits for an ACK (acknowledge to use the message) from the destination process;

If the message is asynchronous, the process that sent SND continues executing the part program without waiting for the message to resume the SND destination process.

IMPORTANT

A process (e.g. process 1) cannot send another process (e.g. process 2) more than one message at a time, at least not until process 2 exits from WAIT and the message is cleared from the queue. However, the sender can send messages to other processes (e.g. process 3).

IMPORTANT

The message queue may be cleared by resetting the process or enabling another part program.

Exchanging data

Commands SND and WAI permit the exchange between processes of data, such as local or system variables, constants, strings.

The data is sent to (SND) or received (WAI) from the processes in question and may be specified in programming of the respective functions; for further details, see the syntax of the two three-letter codes on the following pages.

Resetting synchronised processes

If a reset is made, it must be made both on the processes sending the synchronisation messages (SND) and on the processes receiving the synchronisation messages (WAI).

This is to avoid unwanted messages remaining in the "MESSAGE QUEUE" of the processes, which could lead to malfunctionings.

Channels table

The channels table is a internal system table, containing the identifiers of the logic *channels* on which a process can operate, where *channel* is taken to mean a remote or local environment.

Each process has its own table, each of which contains 40 *channels*. When the system is powered on, it defines by *default* the 20 local processes as the first 20 *channels* of the table, identifying them with numbers 1,2,3,4.....20.

In the event of 10 Series systems connected in a network through the option MINI DNC E65/E66 or higher, other channels may be defined in the table by associating the identifiers (1÷40) with remote environments previously configured with the option MINI DNC.

If a value from 1 to 20 was used as the identifier of a remote environment, the corresponding local process is deleted from the channels table.

DCC - Definition of the communication channel

The three-letter code DCC allows the programmer to define the channels of the table of the process it is programmed in.

Syntax

(DCC ,*channel number***,** *channel name***)**

where:

channel number	Number of the channel being defined; may be a number or a local or system variable, of value between 1 and 40.
channel name	Name of the remote environment defined with the network configurator of the MINI DNC option.

Characteristics:

The *channel* defined with this command must be configured previously with the MINI DNC option. Among the types of channel that may be defined with MINI DNC, command DCC can operate on two predefined types.

- Type 1 - PROCESS channel:
  A remote process to which all the commands accepted by local processes can be sent.

  Examples:

  ```
  (DCC,25,remote_1)  ;defines channel n° 25
  (WAI,P25)          ;wait for message from channel (remote process) n°25

  (PRO,25)
  (EXE,PROGRAM)      ;run part program on remote process n°25
  ```

- Type 2 - PLUS channel:
  A channel on which reading and writing of PLUS variables (SN, SC, L, @, $ASSET) can be rerouted with a PVS command.

IMPORTANT

A RESET of the process or activation of a part program resets the channels table:
channel 1 = local process 1
channel 2 = local process 2
channel 3 = local process 3
channel 4 = local process 4

NOTE:

Each command issued must be consistent with the type of channel it is sent to.
For example, if a command accepted from the process channels (e.g. WAI, SND,EXE) is sent to a channel type 2, the system returns an **NC221 Wrong process type** error.

PVS - PLUS channel selection

Three-letter code PVS permits reading and writing of PLUS variables to be rerouted to a type 2 remote channel, previously defined with the DCC command.

Syntax

(PVS ,*channel number***)**

where:

channel number	Number of the channel being defined; may be a number or a local or system variable, of value between 1 and 40.

Characteristics:

After programming the PVS command, reading and writing of the PLUS variables (SN, SC, L) and @xxxx, $xxxx (with ASSET option enabled) occurs on the remote channel selected.

To restore local PLUS variable reading, program (PVS,0).

If the channel specified is not type 2 (defined with the network configurator), the system returns error: **NC221 Wrong process type**.

If the channel specified was not defined previously with DCC, the system returns error: **NC220 Process undefined**.

IMPORTANT

A RESET of the process cancels programming of the PVS command (tantamount to programming (PVS,0)).

PRO - Definition of the process

The PRO command allows the programmer to specify the process synchronisation commands refer to. The process number is an optional parameter in all commands of this type.

Syntax

(PRO, *process number***)**

where:

process number	It can be a number or a system or local variable in the 1-20 range (1-40 with E65/E66 option). If it is omitted the system will select the default process.

IMPORTANT

A reset of the system or activation of a part program resets the previous definitions of the default process.

SND - Send a synchronisation message

SND allows to send a synchronisation message and data to the specified process.

Syntax

(SND,*[,P process number], S|A [,data] [,data] ....[,data]***)**

---

where:

process number	Destination process number (from 1 to 20, or 1 to 40 with the E65/E66 option). It can be a number, a local or system variable. If it is the number of the process the SND command is executing, an **NC222 Wrong process number** message will be generated. If it is omitted, the information will be sent to the default process declared with PRO.
S | A	S and A specify the two alternative execution modes: **S** (synchronous) the sender or current process goes on WAIT until the destination process has not received the message, cancelled it from the queue and resumed part program execution. **A** (asynchronous) part program execution begins immediately after the message has been sent.
data	The data (optional and max. 20 items) may be Long Real numbers, strings between quotes, and local or system variables. The system reads the programmed data and transmits them to the specified process together with the synchronisation message. It may send up to 20 parameters, which may occupy up to 174 bytes. The length of the various data types is as follows:

Data type	Length (in bytes)
Boolean / Byte	2
Short	3
Long / Real	5
Double (Long Real)	9
String	number of characters + 2

Example:
(SND,P1,S,E1,SN12,"TOOL",SC0.30,@BOOL_PLUS(0),33.6)

The number of bytes transmitted by this block is:

Parameter	Format	Length (in bytes)	
E1	Long Real	9	
SN12	Long Real	9	
"TOOL"	String	6	
SC0.30	String	32	
@BOOL_PLUS(0)	Boolean	2	
33.6	Long Real	9	
	Total:	67	transmitted bytes

The system generates the **NC224 Data sending too long** error if there is an attempt to send more than 174 bytes.

- **Error situations:**

  The **NC225 Data loading failed** error message occurs in two cases:

- *Example 1 (SND synchronous mode)*

  Process 1 sends a synchronous (SND,P2,S,E1) to process 2 while it is on WAIT (WAI,P1,SC0.30):

  - Both processes will display the **NC225 DATA LOADING FAILED** error message and their programs will remain on WAIT.

- *Example 2 (SND asynchronous mode)*

  Process 1 sends the asynchronous (SND,P2,S,E1) to process 2, in which a (WAI,P1,SC0.30) is programmed.
  In this case, as the data is not consistent with that of process 2, and the SND is asynchronous, the result is that:

  - process 1 resumes part program execution
  - process 2 displays the **NC 225** error message and its part program is not allowed to start.

WAI - Wait for a synchronisation message

WAI puts the process on WAIT for a synchronisation message sent by a specified process. If the message is available in the message queue, part program execution continues and the receiver responds to the sender task no matter whether the synchronous SND has been executed or not.

Syntax

(WAI,*[,P process number], [,variable] [,variable] ....[,variable]***)**

---

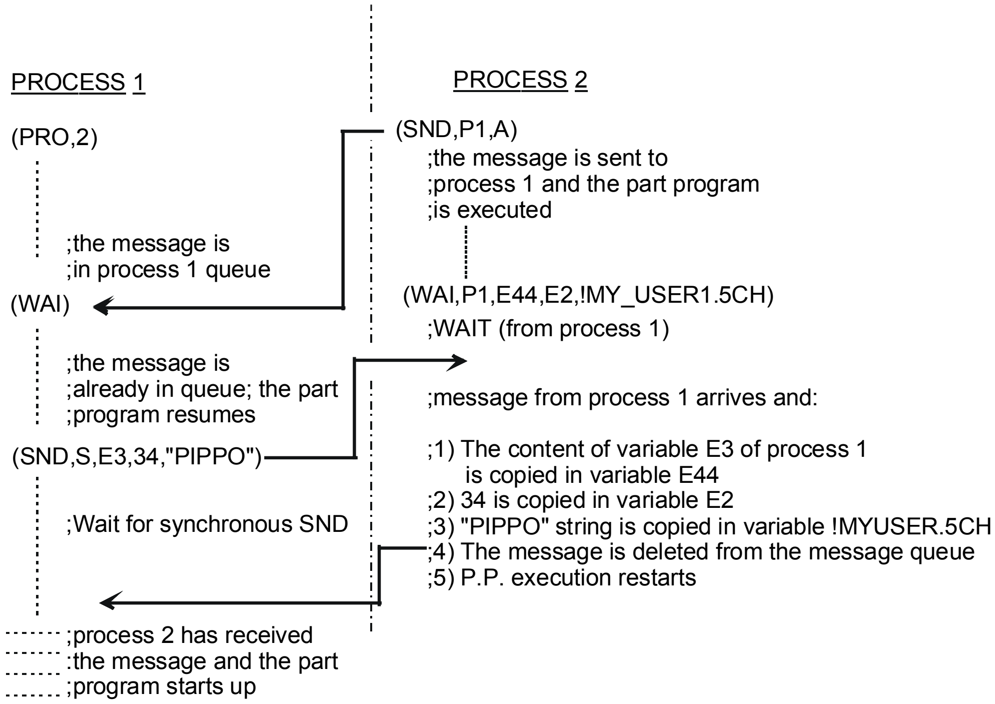
where:

process number	Sender process number (from 1 to 20, or 1 to 40 with the E65/E66 option). It can be a number, a local variable or a system variable. If it is the number of the process the WAI command is executing, a **NC222 Wrong process number** message will be generated. If it is omitted, the information will be sent to the default process declared with PRO.
variable	List of the variables, maximum 20, in which to receive the alphanumeric values arriving with the synchronisation message. If number or type of the variables programmed is not consistent with those sent, the sender process is sent error message **NC255 Data loading failed** (see also the SND command for further details).

IMPORTANT

When a process is on WAIT and a synchronisation message with data arrives, the ACK requested by the sender is sent after the data have been written in the programmed variables. If data fail to be written because of format or process number errors, a negative ACK is sent to the sender.

The figure below illustrates how SND and WAI work.



```
(PRO,2)

    ;the message is
    ;in process 1 queue
(WAI)

    ;the message is
    ;already in queue; the part
    ;program resumes
(SND,S,E3,34,"PIPPO")

    ;Wait for synchronous SND

    ;process 2 has received
    :the message and the part
    ;program starts up
```

```
(SND,P1,A)
    ;the message is sent to
    ;process 1 and the part program
    ;is executed

(WAI,P1,E44,E2,!MY_USER1.5CH)
    ;WAIT (from process 1)

    ;message from process 1 arrives and:

    ;1) The content of variable E3 of process 1
        is copied in variable E44
    ;2) 34 is copied in variable E2
    ;3) "PIPPO" string is copied in variable !MYUSER.5CH
    ;4) The message is deleted from the message queue
    ;5) P.P. execution restarts
```

Example of using SND and WAI commands

EXE - Automatic part program execution

EXE allows to start execution of a part program in the specified process.

Syntax

(EXE,*part program number[,P process number]***)**

---

where:

part program name	Name of the part program to be executed in AUTO mode. It may be up to 48 characters long. It may be a string of characters (not between quotes) or a character variable preceded by ?.
process number	Number of the process (from 1 to 20, or 1 to 40 with E65/E66 option) that runs the program. It can be a number, a local variable or a system variable. If it is the number of the process that is executing the EXE command, an **NC222 Wrong process number** message will be generated. If it is omitted, the information will be sent to the default process declared with PRO.

Characteristics:

The EXE command forces execution of the specified process in AUTO mode, and activates and launches the part program (CYCLE ON).

SETMODE, SPG and CYCLE ON can be filtered by the machine logic.

If SETMODE, SPG or CYCLE ON are received by a process that cannot execute them because it is on RUN or HOLD, the **NC227 EXE or ECM failed** error message will be displayed.

Errors returned by the EXE command can be managed by a part program through the STE and ERR variables. In this connection, see appendix C.

Examples:

```
(EXE, MAIN_PROG ,P3)          activates execution of the MAIN_PROG program by process 3

(PRO,2)
SC1.10="MAIN_2"
EXE,?SC1.10)                   activates execution of the MAIN_2 program by process 2
```

ECM - Manual block execution in a process

ECM allows to manually execute a block in the specified process.

Syntax

(ECM,*command*[,P *process number*] [,S]**)**

---

where:

command	Command code. It may be a string or a string variable between apexes or quotes. The apexes are used when in the command there are quotes. **Example:** (ECM,'(DIS,"PROVA")',P2)
process number	Process where Part Program is activated. Number or the process number from 1 to 20 (1 to 40 with E65/E66 option)that runs the program. It can be a number, a local variable or a system variable. If it is the number of the process executing the EMC command, an **NC222 Wrong process number** message will be generated. If it is omitted, the information will be sent to the default process declared with PRO.
S:	optional parameter forcing synchronous mode operation

Characteristics:

The EMC command forces the destination process in MANUAL mode, sends the specified command to it and executes it via CYCLE ON. Both AUTO and CYCLE ON can be filtered by the machine logic.

The ECM command by default (without the S parameter) is of the asynchronous type: this means that after executing ECM, the part program continues with the following blocks irrespective of whether the command sent via ECM to another process has been concluded or not.

It is therefore important not to execute another ECM command on the same process until the execution of the previous command has been completed.

If this occurs, the "**NC 227 EXE or ECM command failed**" error is signalled.

The ECM command with the S parameter is of synchronous type; this means that after the execution of ECM, before proceeding with the following blocks the part program waits until the command has been completed on the recipient process.

If MDI or CYCLE ON are received by a process that cannot execute them because it is on RUN or HOLD, the **NC227 EXE or ECM failed** error message will be displayed.

Errors returned by the ECM command can be managed by a part program through the STE and ERR variables. In this connection, see appendix C.

Examples:

(ECM,"E1=12",P2)	Sets to 12 the value of local variable E1 in process 2
(PRO,2)	Other example:
SC0.5="E1=12"	Prepares the command in variable SC0.5
(ECM,SC0.5)	Executes ECM for a command written in the content of the variable

Example of synchronisation of two process using EXE:

;N1 SNA_N			;N1 SNB_N
;N2 MULTIPROCESS: SYNCHRONISATION			;N2 MULTIPROCESS; SYNCHRONISATION
N3 (DIS,"PROGRAM SYSTEM 1")			N3 (DIS,"PROGRAM SYSTEM 2")
N4 (EXE,SNB_N,P2)			/N4 (UGS,X,-100,100,Y,-130,150,Z)
N5 (UGS,X,-100,100,Y,-100,100,Z)			N5 SN5=0
N6 T2.2M6			N6 T4.4M6
N7 S1500F500M3		→	N7 (WAI,P1)
N8 XYZ			N8 S400F400M3
N9 E25=0			N9 XY
N10 G81R-14Z-35			N10 E25=0
N11 (RPT,6)			N11 G84R-10Z-22
N12 (ROT,E25)			N12 (RPT,6)
N13 X-80Y			N13 (ROT,E25)
N14 E25=E25+60			N14 X-80Y
N15 (ERP)			N15 E25=E25+60
N16 G80Z50			N16 (ERP)
N17 (ROT,0)			N17 (ROT,0)
N18 (SND,P2,A)	→		N18 G80ZM5
N19 (CLS,MA)		→	N19 (WAI,P1)
N20 (SND,P2,S)	→		N20 GZ40
N21 GZ90		→	N21 (WAI,P1)
N22 (SND,P2,A)	→		N22 E8=0
N23 (WOS,SN5>=9)	←		N23 T2,2M6
N24 S200F500M3			N24 S1300F500M3
N25 X-10Y50			N25 G81R2Z-25
N26 Z-25			"D" N26
N27 G1G41X			N27 G91X10
N28 G2IJ			N28 G90
N29 G1G40X10			N29 E8=E8+1
N30 GZM5		←	N30 #SN5=E8
N31 M30			N31 (GTO,D,E8<10)
			N32 G80M5

SHARED AXES

General

The "shared axes" feature allows one or more processes to acquire or release axes resources. An axis is acquired or released by associating its name (X,Y,Z) to one of the identifiers configured in AMP (from 1 to 32).

- A name cannot be assigned to different axes in a process.
- An identifier cannot be assigned to different machine axes.

Conditions for axis acquisition

Before acquiring an axis check whether the following conditions apply:

- the axis is configured in AMP and the identifier exists in the configured processes;
- the axis has been released from all processes;
- the axis name is specified in the program;
- the identifier (ID) is a number between 1 and 32 and is associated neither to a spindle nor to an auxiliary axis;
- the name to be assigned to the axis is not used for a pseudo axis in a process.

GTA - Axes acquisition

GTA allows one or more axes to shift between processes. In addition, it redefines the axis/offset association and sets the sequence in which axes are displayed.

Syntax

(**GTA,**[*T value*,]*ID_axes_name* [, *ID_axes_name*...] [/] [*axis_1_name*][,*axes_2_name*])

where:

T value	value = 0	loose activate origins (this is the same as omitting the T0 parameter)
	value = 1	maintain active origins on the re-programmed axes (with the same name, ID and order) and on the programmed axes with a different name but the with the same ID and order.
	value = 2:	maintain active origins on all set axes, regardless of the programming origin of the ID's.
/		Is the separator between the axis name and the optional length offset.
axis_1_name		Are the axes to which length offsets 1 and 2 will be applied the next time an
axis_2_name		offset is enabled.

Characteristics:

- In a GTA block you must declare all the controlled axes, including those already belonging to the process.
- If an axis is not specified in a GTA command it will be released and made available to other processes.
- When a GTA command is executed the axes are displayed in the order in which they are specified. GTA can be used for changing this order.
- The RESET command cancels the changes made by GTA.
- At power up axes and processes will be associated as in AMP.
- When the control executes a tool change or enables an offset, the axes to which length offsets are associated must already be controlled by the process.

INHIBITING GTA

Axes acquisition/release with GTA may be inhibited in the following cases:

- when a radius offset is active
- when a canned cycle (G81-89) is active
- when the system is on HOLD or IDLE-MAS
- when a virtual mode is active.
- When the tool correction mode (h or T) is active and you are releasing an axis where the tool correction has been applied or are acquiring a new axis to which the tool correction must be applied.
- Release of axes shared with the logic.

SYSTEM PARAMETERS AFTER GTA

A GTA command permits to initialise the following information:

Interpolation plane

- The interpolation plane is defined by two of the first three axes programmed by the current G (G17, G18, G19). If GTA defines less than three axes, the first and second axes will be forced as abscissa and ordinate.
- To ensure correct operation, the desired plane must be programmed with G16.

Origins

- If the "Tvalue" parameter is omitted or set to "value=0", the active origins will be reset and the RESET ORIGINS logic function is called with the list of axes controlled by the process.
- If the "Tvalue" parameter is set to value=01, the origins will be maintained on the reprogrammed axes with the same name, ID and order and on the programmed axes with a different name but the same ID and order.

Mirror:

- If the “Tvalue” parameter is set as value = 2, the origins will be maintained on all programmed axes, even if their ID’s are not the same.
- Mirroring is reset for all axes

Scale factors:

- Scale factors are reset for all axes

ROT rotation:

- Rotation programmed with a ROT command will be cancelled.

Travely limits:

- Travel limits for all the axes return to configured values.

Axis selection for manual movements:

- After a GTA command, the first displayed axis will be selected for manual move execution.

Length offsets

- The axes identifiers to which length offsets 1 and/or 2 are associated will be programmed as an option on GTA. Default associations defined in AMP will be lost. They will be restored after the control has been switched on again.

INFORMATION RETAINED AFTER A GTA COMMAND

- **Locked axes**
- **Plane rotation activated with through a manual PROGRAM SETUP command**
- **Incremental and provisional origins**

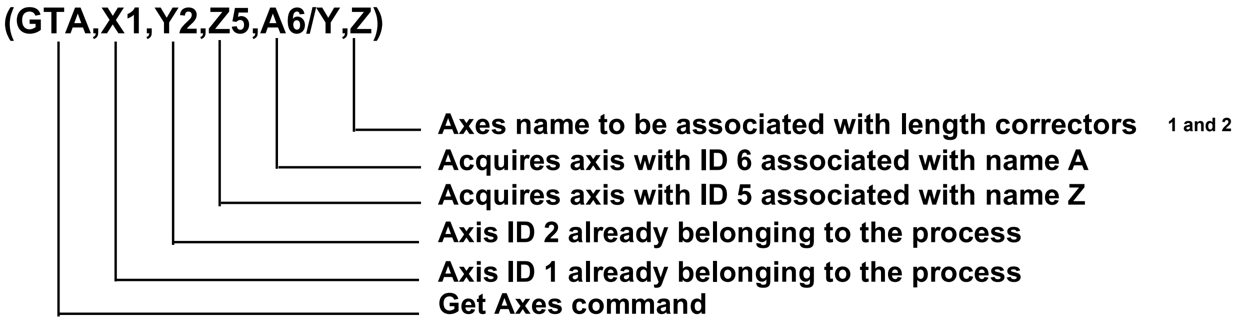
Example 1:
Initial conditions:

- Controlled axes are X (with ID 1) and Y (with ID 2).

Goals:

- Maintain control over axes X and Y (ID 1 and 2)
- Acquire axes associated to ID 5 and 6 and name them Z and A
- Associate length offset 1 to axis Y (ID 2)
- Associate length offset 2 to axis Z (ID 5)
- Obtain the following display order: X, Y, Z, A.

Programming example:


(GTA,X1,Y2,Z5,A6/Y,Z)
Axes name to be associated with length correctors
1 and 2
Acquires axis with ID 6 associated with name A
Acquires axis with ID 5 associated with name Z
Axis ID 2 already belonging to the process
Axis ID 1 already belonging to the process
Get Axes command

IMPORTANT

Programming all the axes (new and old) is mandatory. It permits to reassign axes names and including those already belonging to the process is mandatory and allows to redefine both the axis names and the display order.

Example 2:

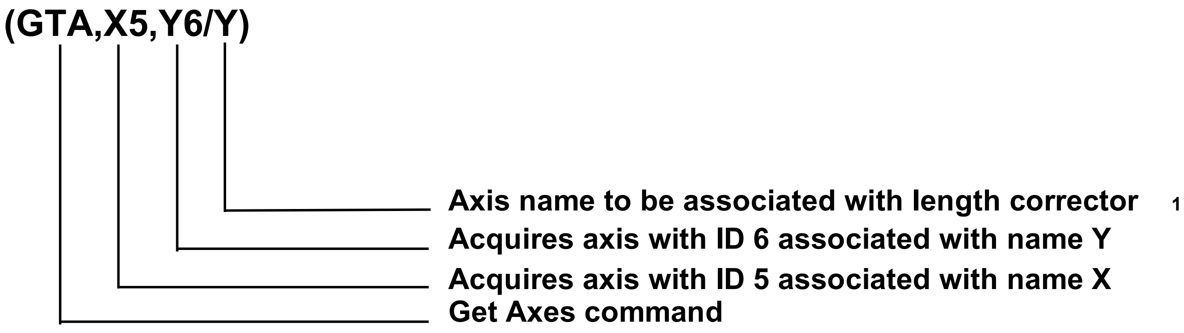
Initial conditions:

- Controlled axes are X (with ID 1) and Y (with ID 2).

Goals:

- Release axes X and Y (ID 1 and 2)
- Acquire axes associated to ID 5 and 6 and name them X and Y
- Associate length offset 1 to axis Y (ID 6)
- Obtain the following display order: X, Y.

Programming example:


(GTA,X5,Y6/Y)
Axis name to be associated with length corrector
1
Acquires axis with ID 6 associated with name Y
Acquires axis with ID 5 associated with name X
Get Axes command

Example 3:

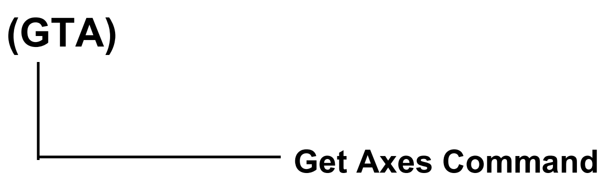
Initial conditions:

- An indefinite number of controlled axes

Goals:

- Release all its axes

Programming example:



Example 4:

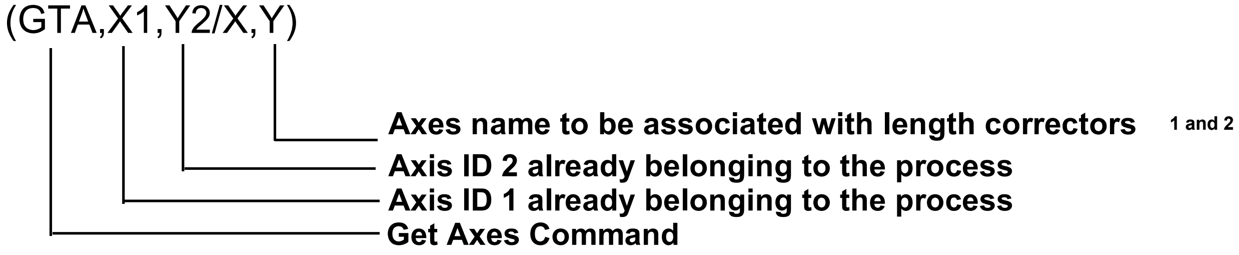
Initial conditions:

- Controlled axes are X,Y,Z,A,B with ID 1,2,3,4,5.

Goals:

- Retain control of X and Y axes with (ID's 1 and 2)
- Release axes Z, A and B (with ID's 3, 4 and 5)
- Associate length offset 1 to axis Y (ID 6)
- Obtain the following display order: X, Y.

Programming example:


(GTA,X1,Y2/X,Y)
Axes name to be associated with length correctors
1 and 2
Axis ID 2 already belonging to the process
Axis ID 1 already belonging to the process
Get Axes Command

Example 5:

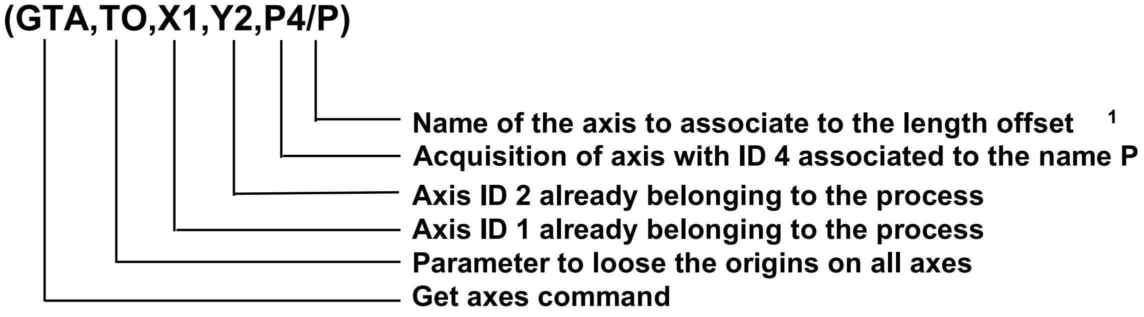
Initial Conditions:

- Controlled axes are X,Y,Z axes with ID 1,2,3.
- To the X,Y,Z axes, the 1 origin is associated.

Goals:

- Retain control of the axes with ID 1 and 2 (X,Y).
- Release the axis Z (ID 3).
- Acquire the P axis with ID 4.
- ssociate length offset 1 with the P axis (ID 4).
- Obtain the following display order: X,Y,P.
- Loose the origin associated to all axes.

Programming example:



```
(GTA,TO,X1,Y2,P4/P)
```

Name of the axis to associate to the length offset 1
Acquisition of axis with ID 4 associated to the name P
Axis ID 2 already belonging to the process
Axis ID 1 already belonging to the process
Parameter to loose the origins on all axes
Get axes command

Example 6:

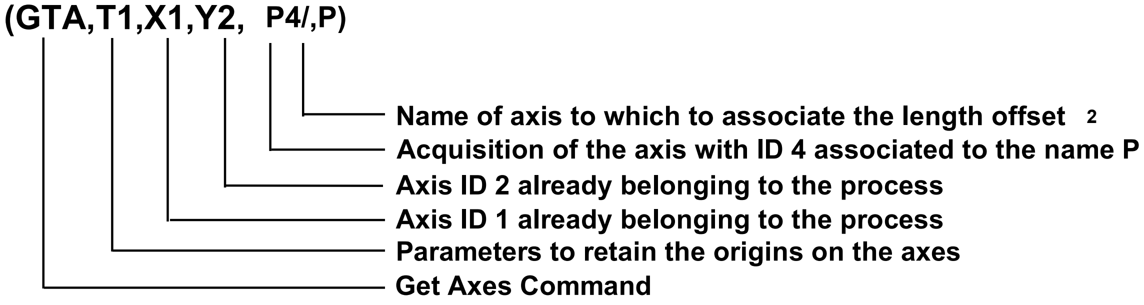
Initial Conditions:

- Controlled axes are X,Y,Z axes with ID 1,2,3.
- To the X,Y,Z axes, 1 origin is associated.

Goals:

- Retain control of the axes with ID 1 and 2 (X,Y).
- Release the axis Z (ID 3).
- Acquire the P axes with ID 4.
- Associate the length offset 2 with the P axis (ID 4).
- Obtain the following display order: X,Y,P.
- Retain the associated origins with the axes with ID 1 and ID 2.

Programming example:



```
(GTA,T1,X1,Y2, P4/,P)
```

Name of axis to which to associate the length offset 2
Acquisition of the axis with ID 4 associated to the name P
Axis ID 2 already belonging to the process
Axis ID 1 already belonging to the process
Parameters to retain the origins on the axes
Get Axes Command

Example 7:

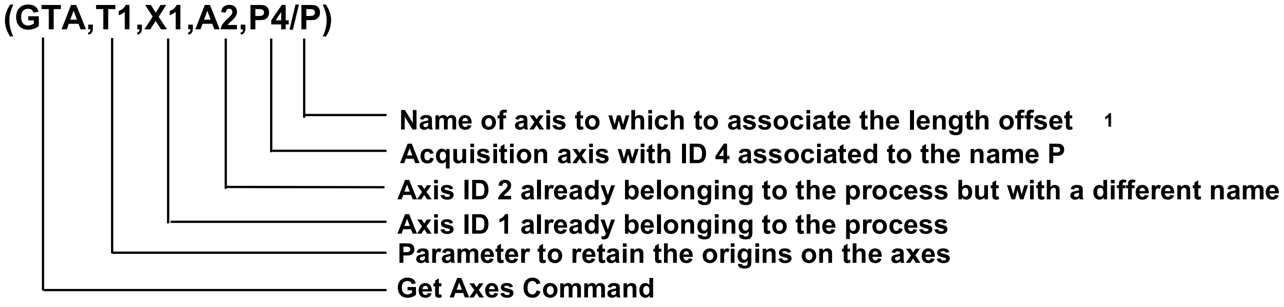
Initial Conditional :

- Controlled axes are the X,Y,Z axes with ID 1,2,3.
- To the X,Y,Z axis 1 origin is associated.

Goals:

- Retain control over the axis with ID 1 (X).
- Retain control over the axis with ID 2 ID 2 (Y) changing the name to (A).
- Release the Z axis (ID 3).
- Acquire the P axis with ID 4.
- Associate the length offset 1 with the P axis (ID 4).
- Obtain the following display order: X,A,P.
- Retain the origin associated with ID 1 and ID 2.

Programming example:



```
(GTA,T1,X1,A2,P4/P)
```

Name of axis to which to associate the length offset 1
Acquisition axis with ID 4 associated to the name P
Axis ID 2 already belonging to the process but with a different name
Axis ID 1 already belonging to the process
Parameter to retain the origins on the axes
Get Axes Command

Example 8:
Initial conditions:

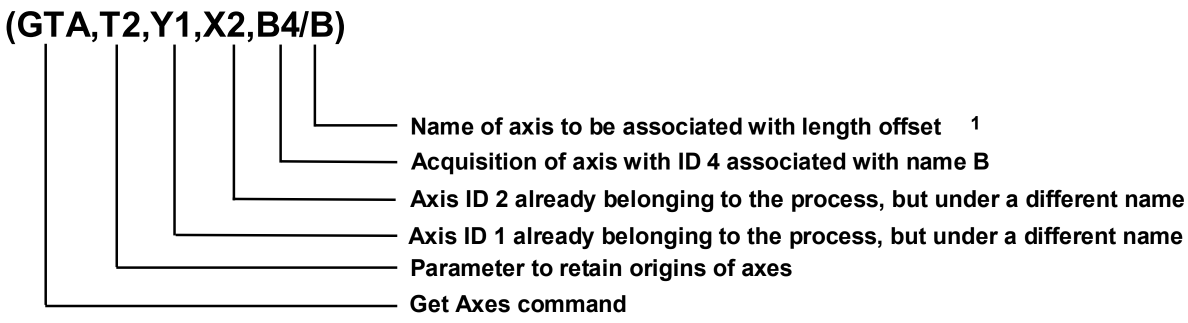
- Controlled axes are X,Y,Z axes with ID 1,2,3.
- Origin 1 is associated with axes X,Y,Z.
- A 2nd process retains axes A, B with ID 4, 5, with which an origin 2 is associated.

Goals:

- Retain control over axes with ID 1, 2 and 4
- Release axis Z (ID 3).
- Acquire axis B, with ID 4, previously released by process 2.
- Associate length offset 1 with axis B (ID 4).
- Obtain the following display order (on system video): Y,X,B.
- Maintain the origins associated with all the axes, i.e. origin 1 for axes with ID’s 1 (Y), 2 (X) and origin 2 for axis with ID 4 (B).

Programming example:

(GTA) on process 2 for the release of A and B



(GTA,T2,Y1,X2,B4/B)

Name of axis to be associated with length offset 1
Acquisition of axis with ID 4 associated with name B
Axis ID 2 already belonging to the process, but under a different name
Axis ID 1 already belonging to the process, but under a different name
Parameter to retain origins of axes
Get Axes command

Example 9:
Initial conditions:

- Controlled axes are X,Y,Z axes with ID 1,2,3,4.
- Axis with ID 4 is “shared with the logic”.

Goal:

- Exchanging the abscissa with the ordinate.

Programming example:

```
(GTA,X2,Y1, Z3,   W4)
```

Axis ID 4 already belonging to the process and shared with the logic

Axis ID 3 already belonging to the process and shared with the logic
Axis ID 1 already belonging to the process, but under a different name
Axis ID 2 already belonging to the process, but under a different name
Get Axes command

NOTE:

An axis shared with the logic cannot be released through a GTA command (the system gives out error message NC124) and, needless to say, since it belongs to the logic, even though it is visible from the process environment, it cannot be programmed.
Hence, if you want to use the GTA command on a process involving one or more axes shared with the logic, you must first reconfirm such shared axes as described in the example.

Error Management

If an error such as axis resource not available, ID axis not found, etc. occurs during part program execution, the programmer may stop the program and display the error or manage it from the program through the ERR and STE variables.

For further information about these variables, refer to Appendix C.

Example of axes acquisition from a process:

```
;N1 SINUS_AM
;N2 ESECUZIONE SU PROC.1
;N3 ACQUISIZIONE ID9-ID10 (NUOVI ASSI A-B) DA PROC. 2
N6 (EXE,GTA1_AM,P2)
N7 (DLY,0.1)
N8 (GTA,X1,Y2,Z3,A9,B10,-Z)
N9 T1.1M6
N10 S1000M3
N11 G0G90XYZAB
N12 G1G91G28
/N13 (UGS,1,Z,-80,80,Y,-50,50,Z)
N14 E25=0.03
N15 X0.2713Y0.0947Z-0.1852A-0.1815B0.1007tE25
N16  X0.2711Y0.0886Z-0.1887A-0.1776B0.1066tE25
N17 X0.2708Y0.0825Z-0.1922A-0.1736B0.1125tE25
N18 X0.2703Y0.0764Z-0.1955A-0.1696B0.1184tE25
N19 X0.2697Y0.0702Z-0.1987A-0.1654B0.1241tE25
N20 X0.2689Y0.0640Z-0.2017A-0.1610B0.1299tE25
..................................
..................................
..................................
N2010 X0.2697Y0.1241Z-0.1654A-0.1987B0.0702tE25
N2011 X0.2703Y0.1184Z-0.1696A-0.1955B0.0764tE25
N2012 X0.2708Y0.1125Z-0.1736A-0.1922B0.0825tE25
N2013 X0.2711Y0.1066Z-0.1776A-0.1887B0.0886tE25
N2014 X0.2713Y0.1007Z-0.1815A-0.1852B0.0947tE25
N2015 G0
N2016 (SND,P2,S)
N2017 (GTA,X1,Y2,Z3)
N2018 M30
```

END OF CHAPTER

Chapter 15

HIGH LEVEL GEOMETRIC PROGRAMMING (GTL)

10 Series CNC systems permit to program a geometric profile on the plane using either the standard programming language (G1-G2-G3) or GTL, a high level programming language.

GTL makes it possible to use the information provided by the drawing in order to program a profile made up of straight lines and circles. The 10 Series CNC calculates tangency and intersection points between these elements.

GTL and standard programming codes can be used in the same program but not in the same profile. GTL requires absolute programming mode (G90).

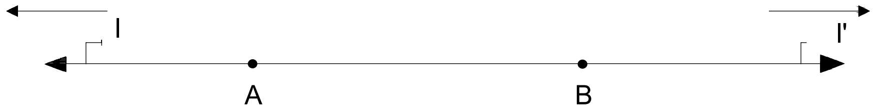
ORIENTED GEOMETRY

With GTL a profile definition is based on four geometric elements:

- reference origins
- points
- straight lines
- circles

A profile is defined not only by its geometric elements but also by a direction of motion. GTL defines the profile elements according to an oriented geometry. For each element you must provide its coordinates on the plane as well a direction.

For example, through points A and B pass two straight lines: I', which moves from A to B, and I, which moves from B to A.



Straight lines I and I'

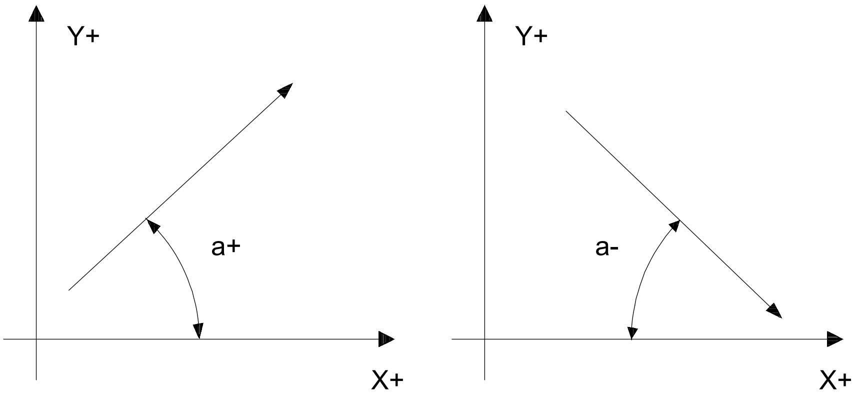
In an oriented geometry I and I' are two different lines.

GTL programming, which is based on oriented geometry, requires the assignment of an assumed direction for each line

IMPORTANT

By convention the direction of a line is given by the angle formed between the line and the positive X semiaxis.

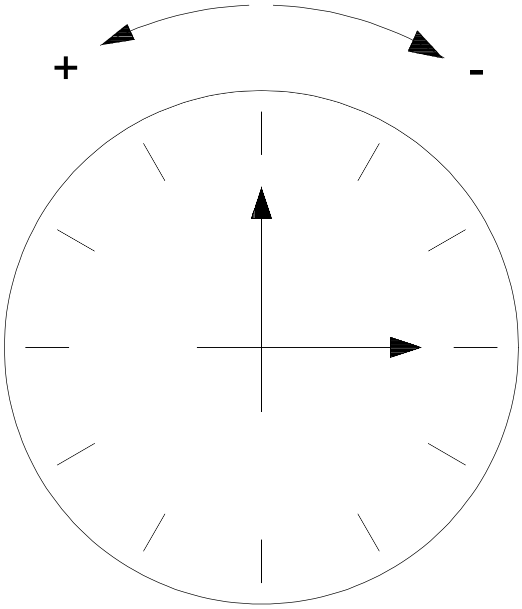
To obtain the angle rotate the positive X semiaxis until it has the direction of the straight line to be defined. The angle will be positive if the positive X semiaxis rotates counterclockwise; it will be negative if the positive X semiaxis rotates clockwise.



Conventional angle signs

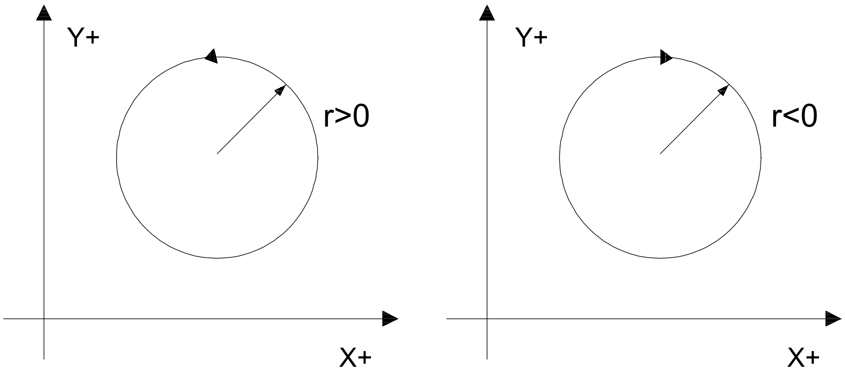
IMPORTANT

Circle definitions must also include a direction of motion. By convention, a counterclockwise motion defines a positve circle and a clockwise motion defines a negative circle.



Clockwise and counterclockwise circles

The radius of a positive circle has a positive sign whereas the radius of a negative circle has a negative sign.



Circle radius signs

If necessary you can reassign the direction of an element during profile definition.

DEFINING GEOMETRIC ELEMENTS

GTL uses lower case letters a-l-c-d-m-o-r-p-s-b to define, respectively, angles, straight lines, circles, distances, modules. reference origins, radiuses, points, intersection numbers, and bevels.

Lower cases are mandatory, because the NC language uses the same letters in capitals to designate other parameters.

Geometric elements must be saved in the program before defining the profile.

GTL straight lines, circles, points, reference origins are geometric variables identified by a NAME and an INDEX. The element must be defined in an assignment block.

The block format is as follows:

NAME INDEX = <Expression>

where

NAME is one of the symbols for geometric element:

o	defines reference origins
p	defines points
l	defines straight lines
c	defines circles

INDEX defines the number of the **NAME** geometric variable (element).
It is a number ranging from 0 to the maximum configured limit.

expression provides all the information that defines the geometric element.

Geometric elements may be defined:

- **directly (explicitly)**, by writing in a block all the necessary information about the geometric element;
- **indirectly (implicitly)**, by writing in a block other geometric elements specified in previous definitions.

Examples of element definitions:

```
o1  =  X30 Y30 a45
p1  =  o1 X15 Y15
p2  =  X60 Y30
l1  =  p1, p2
l2  =  X30 Y50, a45
c1  =  l1,l2,r15
l3  =  X0 Y0, X100 Y60
p3  =  l3, c1
c2  =  p3,r8
................
```

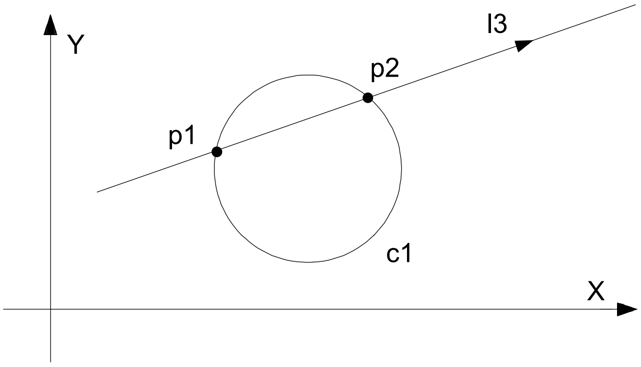
The maximum number of geometric elements must be specified in the system configuration.

The syntax of a geometric definition includes a comma (",") as separator between two geometric elements (straight line - point - circle) or between a geometric element and the subsequent parameter ("r" radius or "a" angle).

Examples:

```
p1    =  X30 Y30          the separator must be omitted
c1    =  I10 J20 r30

l1    =  X20 Y20, X100 Y-10
             |          |
           point      point

l2    =  I30 J30 r10, X80 Y80
             |       |
           circle   point

l3    =  X100 Y100, a45
              |      |
            point   angle

p5    =  c1, c2
          |   |
          |  circle
         circle

c3    =  l1, l2, r18
          |   |   |
          |   |radius
          |straight line
         straight line
```

s2 is a discriminator that selects the second intersection (s2).



Intersection points between straight lines and circles

p1 = c1, l3
p2 = c1, l3, s2

GEOMETRIC DEFINITIONS

Reference origin	on = X.. Y.. a..

Points	pn = [on] X.. Y..
	pn = [on] m.. a..
	pn = ln, ln'
	pn = [-] ln, cn [,s2]
	pn = cn, [-]ln [.s2]
	pn = cn, cn [,s2]

Straight lines	ln = [on] X.. Y.., [on] X.. Y..
	ln = [on] X.. Y , a..
	ln = [on] I.. J.. r.., [on] I.. J.. r..
	ln = [on] I.. J.. r.., a..
	ln = [on] I.. J.. r.., [on'] X.. Y..
	ln = [on] X.. Y.., [on'] I.. J.. r..
	ln = pn, pn'
	ln = pn, a..
	ln = [-]cn, [-]cn'
	ln = [-]cn, a..
	ln = [-]cn, pn'
	ln = pn, [-]cn'
	ln = [-]ln, d..

Circles	cn = [on] I.. J.. r..
	cn = [on] m.. a.. r..
	cn = [-]ln, [-]ln, r..
	cn = [-]ln, [-]cn, r..
	cn = [-]cn, [-]ln, r..
	cn = pn, [-]ln, r..
	cn = [-]ln, pn, r..
	cn = [-]cn, [-]cn', r..
	cn = pn, [-]cn, r..
	cn = [-]cn, pn, r..
	cn = pn, pn', r..
	cn = pn, [-]ln
	cn = pn, [-]cp [,s2]
	cn = pn, pn', pn''
	cn = pn, r..
	cn = [-]cn, [-]d..

NOTE:

Two dots in a definition idicate that a numerical value is mandatory. Parameters enclosed between square brackets [ ] are optional and may be omitted.

DEFINITION OF A REFERENCE ORIGIN

Function	GTL permits to define reference origins using a direct (explicit) format.
Description	The information provided by the program typically describes one system of axes that correspond to the machine axes. For specific design purposes, however, the part may have been drawn according to other cartesian systems, such as the absolute system or other origins obtained by translating or rotating the axes. The GTL geometry may be defined for any system of axes.

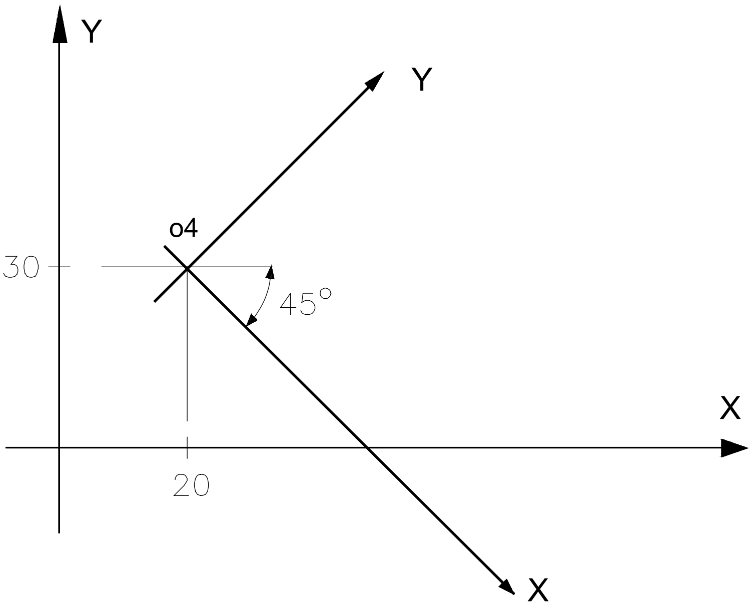
Direct format:

on = X.. Y.. a..

where

on	identifies the name of the reference origin
X.. Y..	coordinates of the new origin
a..	rotation angle (positive if counterclockwise).

Example:

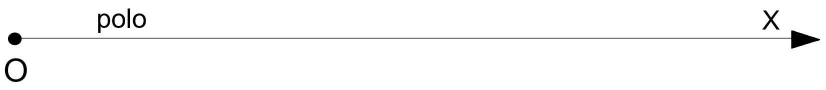


o4 = X20 Y30 a-45

DEFINITION OF POINTS

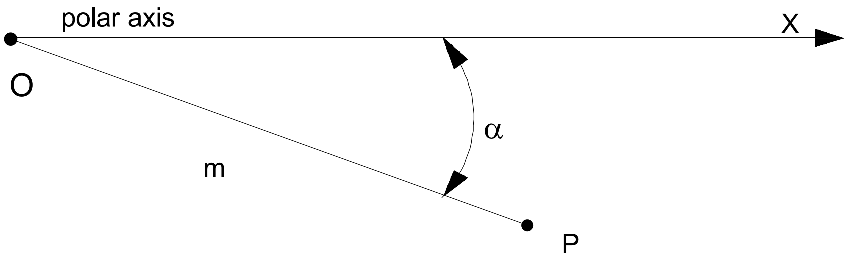
Function GTL permits to define points both directly (explicitly) or indirectly (implicitly). The definition may be written in cartesian or in polar coordinates.

Description The polar reference system consists of an origin (pole) and an X semiaxis also known as polar axis.



Polar axis

Each point on the plane (P) may be defined by its module (i.e. the length of the segment through the point and P) and the value of the angle formed by the polar axis and the P-pole segment.



Polar coordinates

Direct format

Point in cartesian coordinates

pn = [on] X.. Y..

Point in polar coordinates

pn = [on] m.. a..

Indirect format

Intersection point between two predefined straight lines

pn = ln, ln'

Intersection point between a predefined straight line and a predefined circle

pn = [-]ln, cn' [,s2]

pn = cn, [-]ln [,s2]

Intersection point between two circles

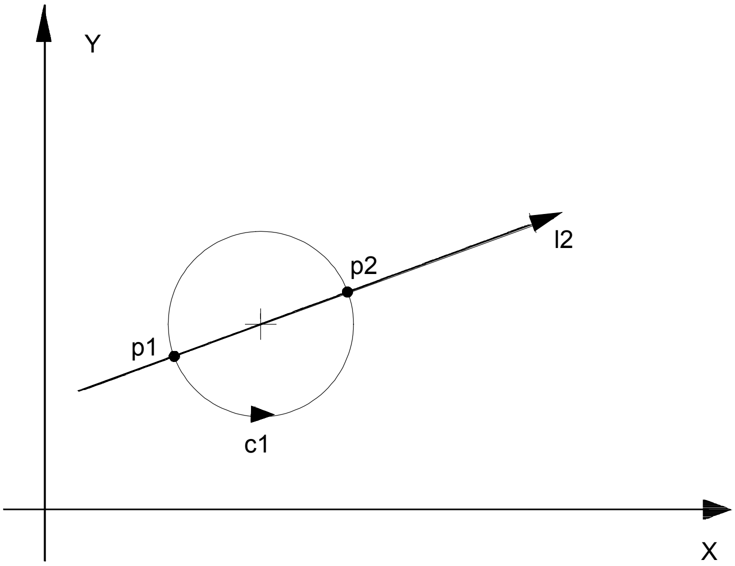
pn = cn, cn' [,s2]

where

pn	is the name of a point whose index (n) is a number ranging from 1 and the configured maximum.
X.. Y..	coordinates of the point
[on]	defined reference origin to whose n index refer the X and Y coordinates
m..	polar vector module
a..	polar vector angle
cn **cn'**	predefined circles with indexes n and n'
[-]ln **[-]ln**	predefined straight lines with indexes n and n'. The direction may be inverted by writing a "-" sign
[,s2]	second intersection discriminator

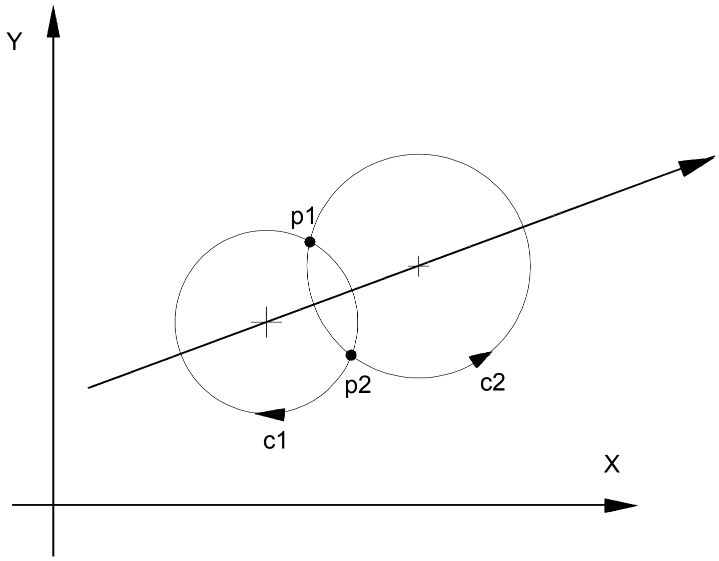
NOTE:

In intersections between straight lines and circles, there are always two possible intersection points. The c1 circle and the l2 straight line intersect in points p1 and p2. If you move along the l2 straight line, the first intersection point is p1 and the second intersection point is p2 . To select the second intersection (p2) use the s2 discriminator. If s2 is omitted, the default intersection point will be the first one (p1).

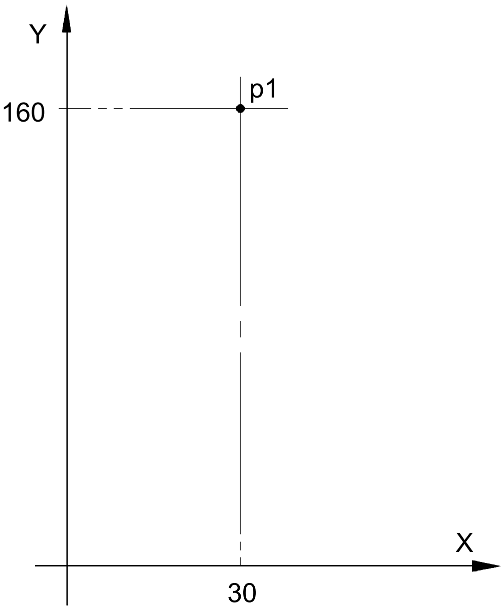


Intersection between a straight line and a circle

In circle-circle intersections there are also two possible solutions. The intersection points between c1 and c2 are p1 and p2. Take the oriented straight line through the centers of the first and the second circles. This line devides the plane into two halfplanes. To select a point from the right hand halfplane (p2) write the s2 discriminator. If s2 is omitted, the default selection will be the point from the left hand halfplane (p1).

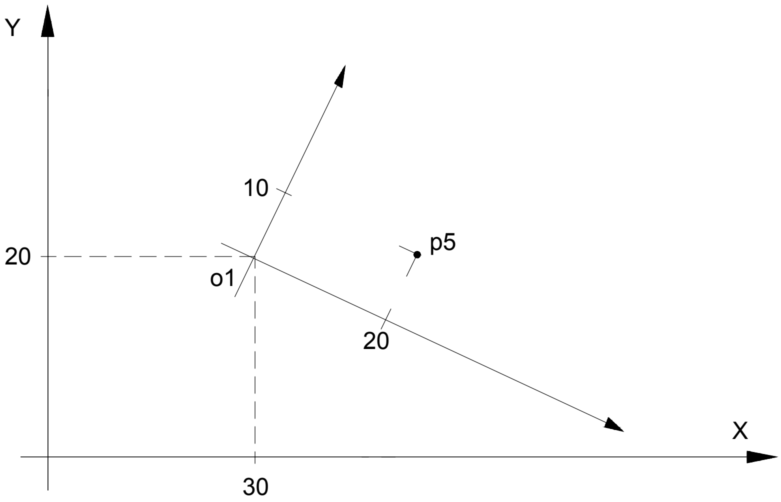


Circle-circle intersection.



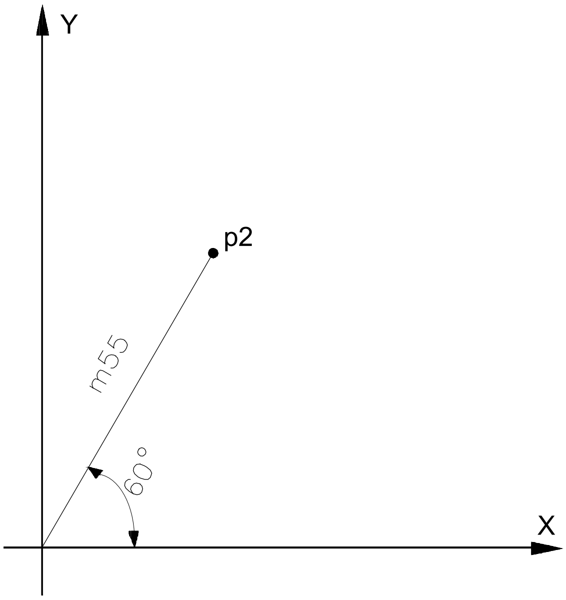
Point in cartesian coordinates

p1 = X30 Y160



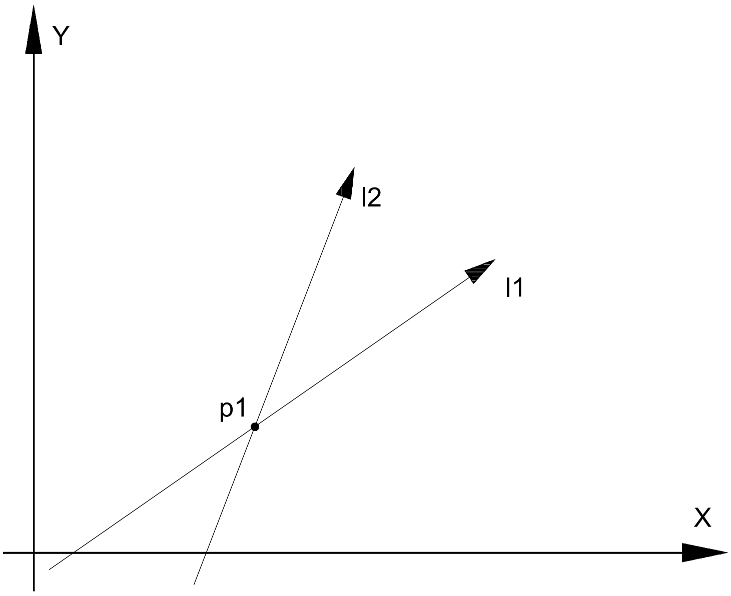
Point in cartesian coordinates

o1 = X30 Y20 a-20
p5 = o1 X20 Y10



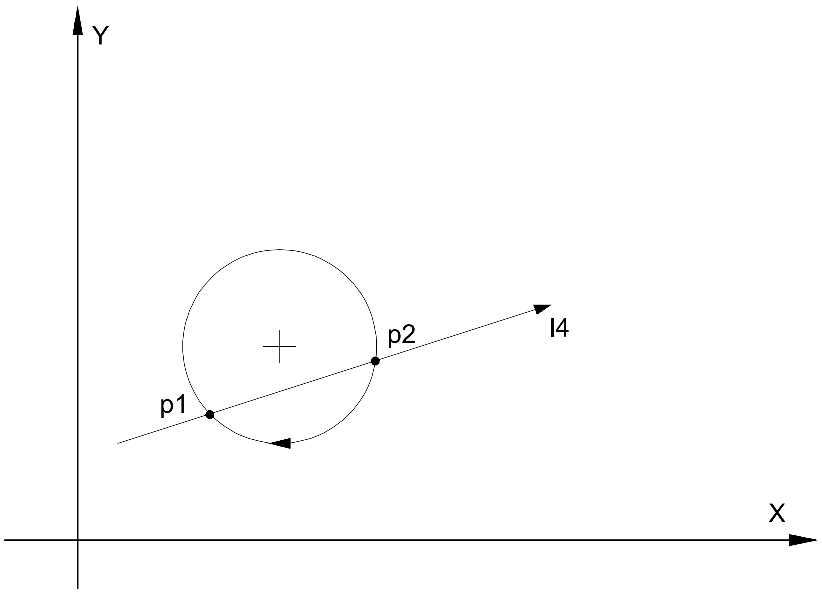
Point in polar coordinates

p2 = m55 a60



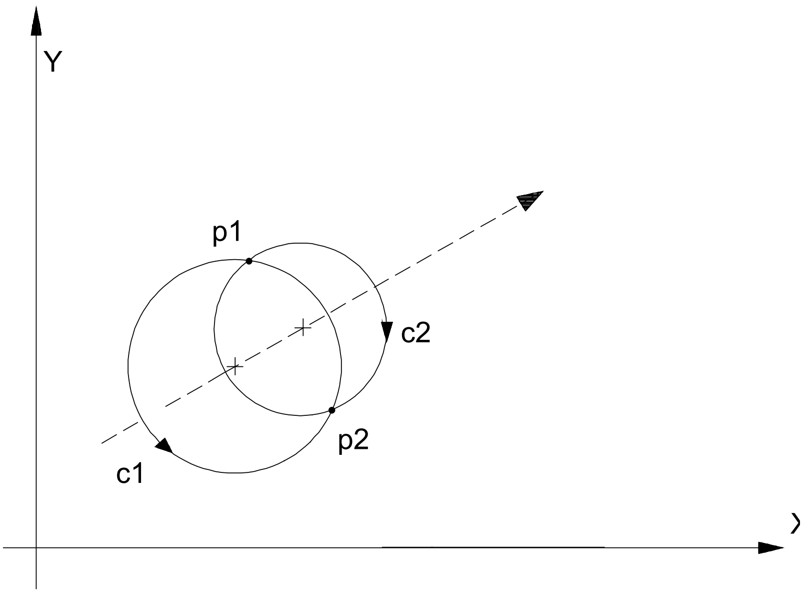
Intersection point between two predefined straight lines

p1 = l1, l2



Intersection points between a straight line and a circle

p1 = l4, c3
p2 = l4, c3, s2
p1 = -l4, c3, s2



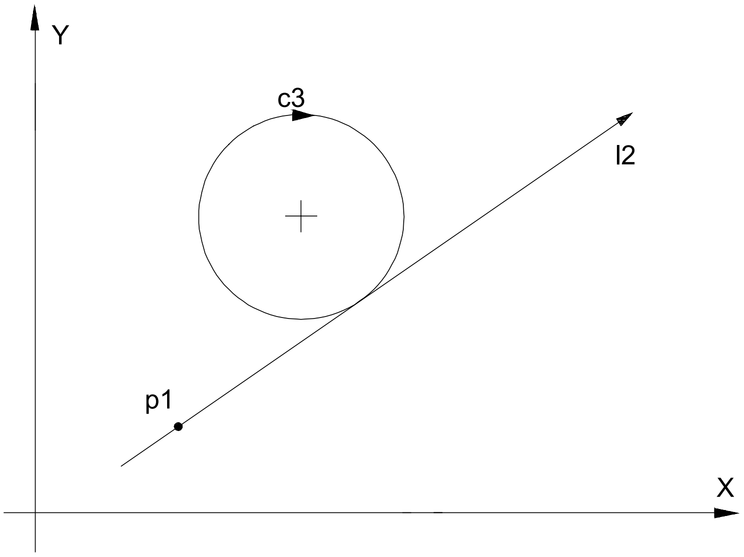
Intersection points between two circles

p1 = c1, c2
p2 = c1, c2, s2
p1 = c2, c1, s2

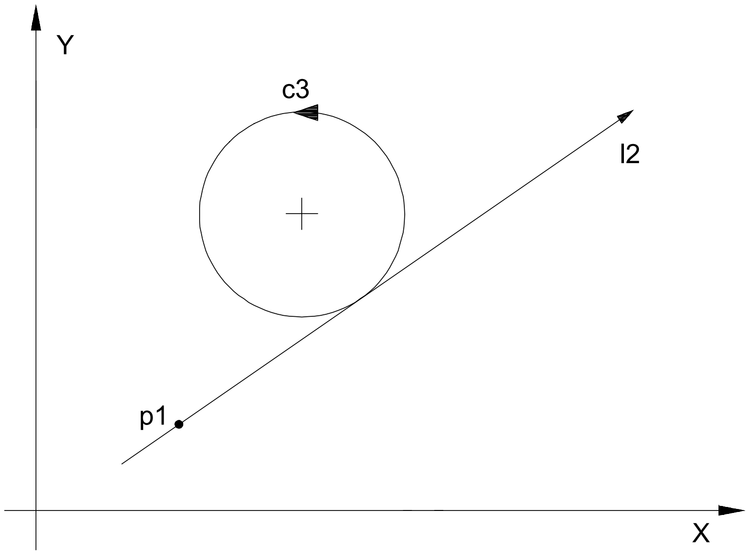
DEFINITION OF STRAIGHT LINES

Function GTL permits to define straight lines both directly (explicitly) or indirectly (implicitly). The definition may be written in cartesian or in polar coordinates.

Description The straight line is directed from the first to the second defined element. If the straight line is tangent to a circle there are two possible solutions because the straight line may be tangent to either side of the circle. To select one solution check that the circle and the straight line have the same direction in the tangency point.



Incompatible directions between two geometric elements



Compatible directions between two geometric elements

Direct format

Straight line through two points

ln = [on] X.. Y.., [on'] X.. Y..

Straight line through a point that forms an angle with the abscissa axis:

ln = [on] X.. Y.., a..

Straight line tangent to a circle that forms an angle with the abscissa axis:

ln = [on] I.. J.. r.., a..

Straight line tangent to two circles:

ln = [on] I.. J.. r.., [on'] I.. J.. r..

Straight line tangent to a circle and through a point:

ln = [on] I.. J.. r.., [on] X..Y..
ln = [on] X.. Y.., [on] I.. J.. r..

Indirect format

Straight line through two points:

ln = pn, pn'

Straight line through a point that forms an angle with the abscissa axis:

ln = pn, a..

Straight line tangent to two circles:

ln = [-]cn, [-]cn'

Straight line tangent to a circle that forms an angle with the abscissa axis:

ln = [-]cn, a..

Straight line tangent to a circle and through a point:

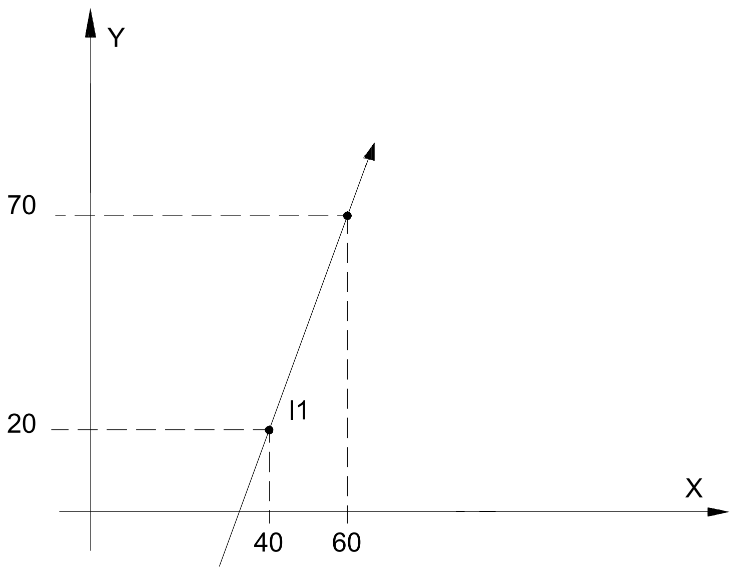
ln = [-] cn, pn
ln = pn, [-]cn

Straight line parallel to a straight line at d distance:

ln = [-]ln, d..
ln = [-]cn, a..

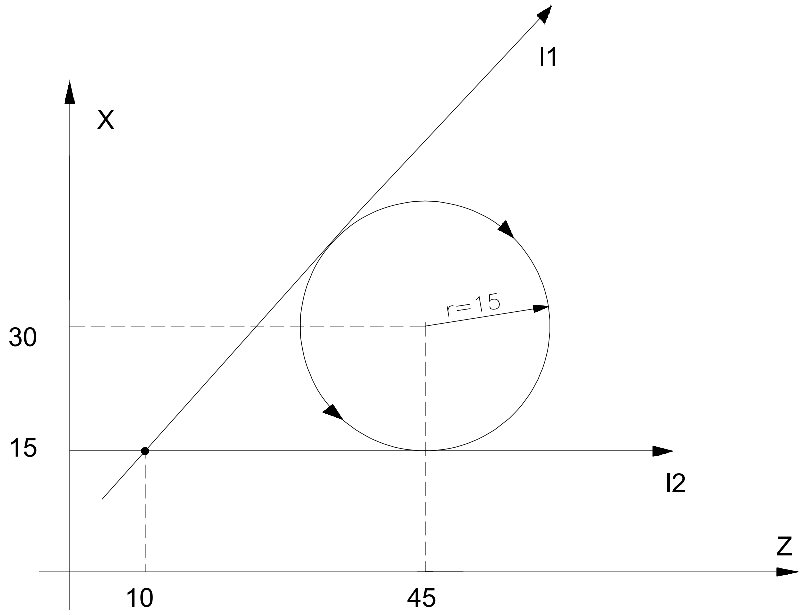
where

ln	identifies the name of the straight line whose index n is a number from 0 through the configured maximum
X.. Y..	point coordinates
a..	angle formed by the abscissa axis and the straight line (positive if counterclockwise)
r..	circle radius (positive if counterclockwise)
pn pn'	predefined points with indexes n and n'
[-]cn **[-]cn'**	predefined circles with indexes n and n'. To ensure compatibility between the direction of the straight line and that of the circle in the tangency point, you may write the - sign to change the direction of the circle.
[-]ln	predefined straight line with n index
d..	distance between two straight lines. It is positive if looking in the direction of the predefined straight line the straight line is on the left hand side. It is negative if looking in the direction of the predefined straight line the straight line is on the right hand side.



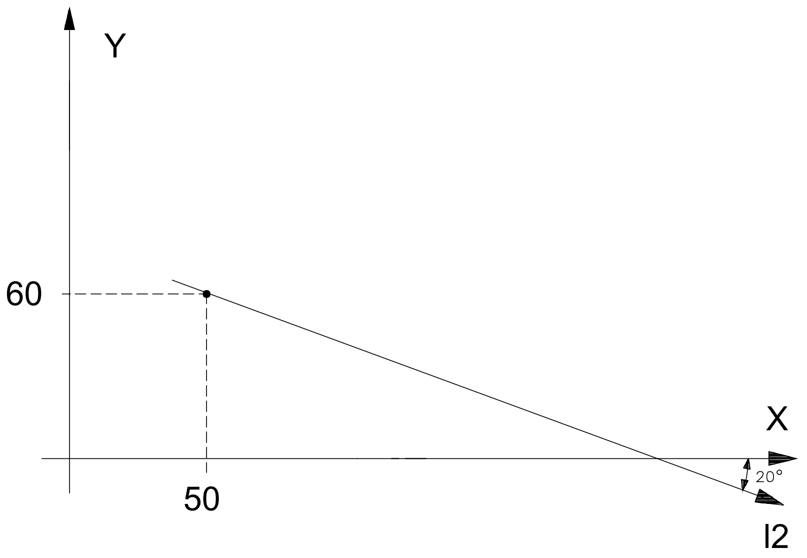
Straight line through two points

I1 = X40 Y20, X60 Y70



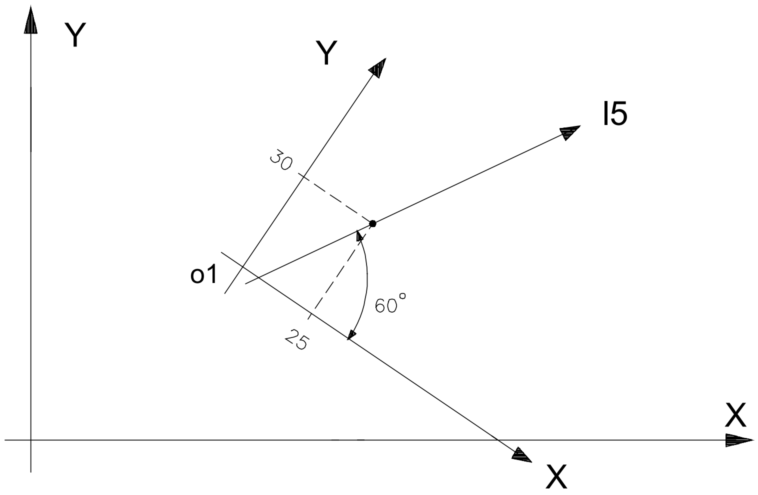
Straight line tangent to a circle
and passing through a point

I1 = X10 Y15,I45 J30 r-15
I2 = X10 Y15,I45 J30 r15



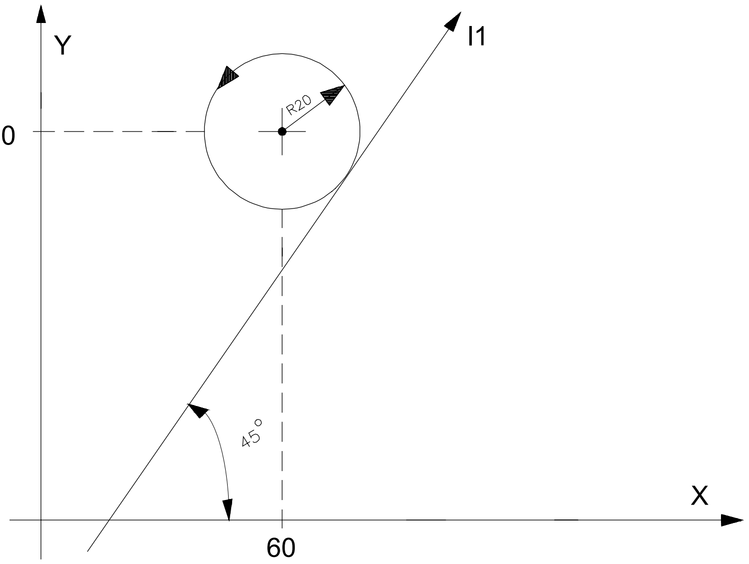
Straight line through a point and forming an angle with the abscissa axis

I2 = X50 Y60, a-20



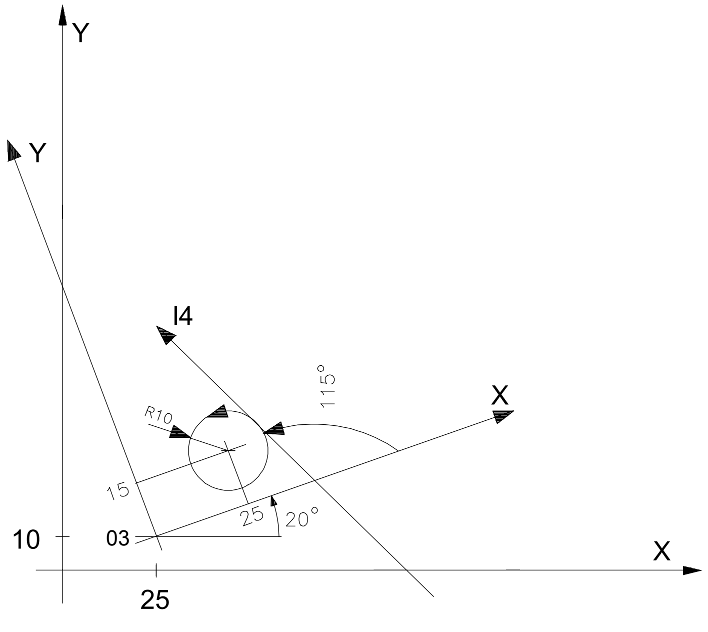
Straight line through a point and forming an angle with the abscissa axis

I5 = o1 X25 Y30, a60



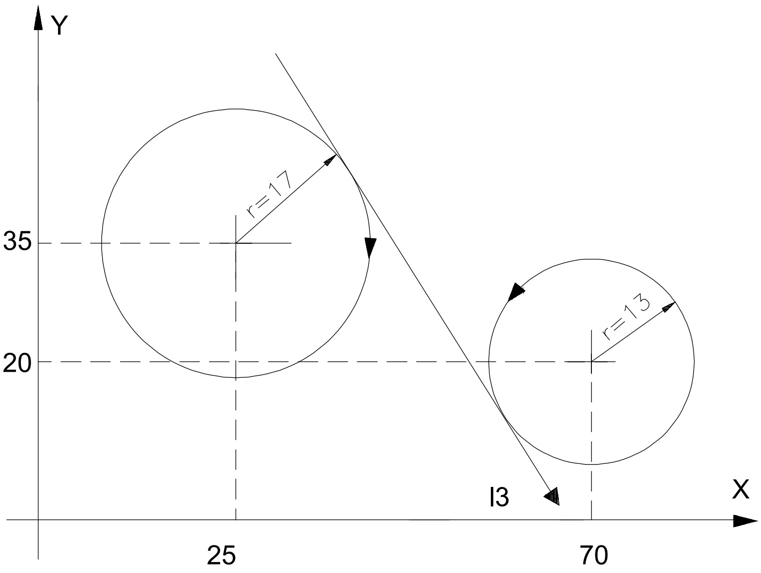
Straight line tangent to a circle and forming an angle with the abscissa axis

I1 = I60 J80 r20, a45



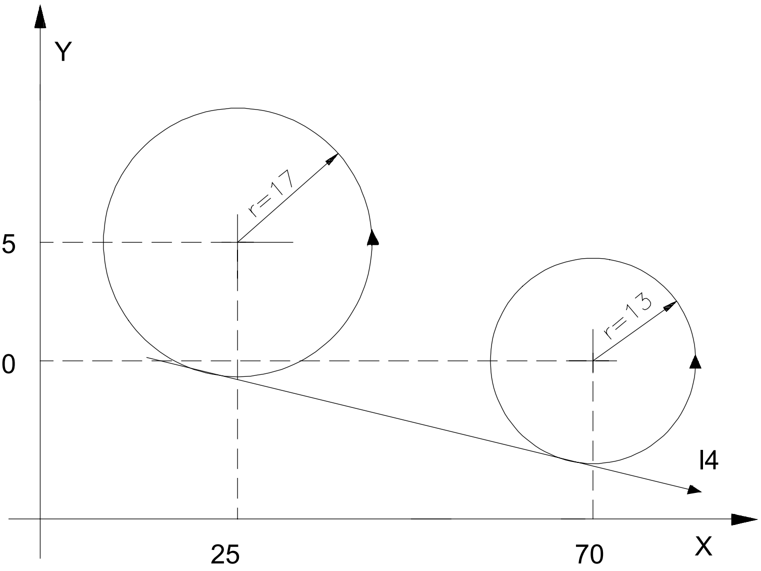
Straight line tangent to a circle and forming an angle with the abscissa axis

I4 = o3 I25 J15 r10, a115



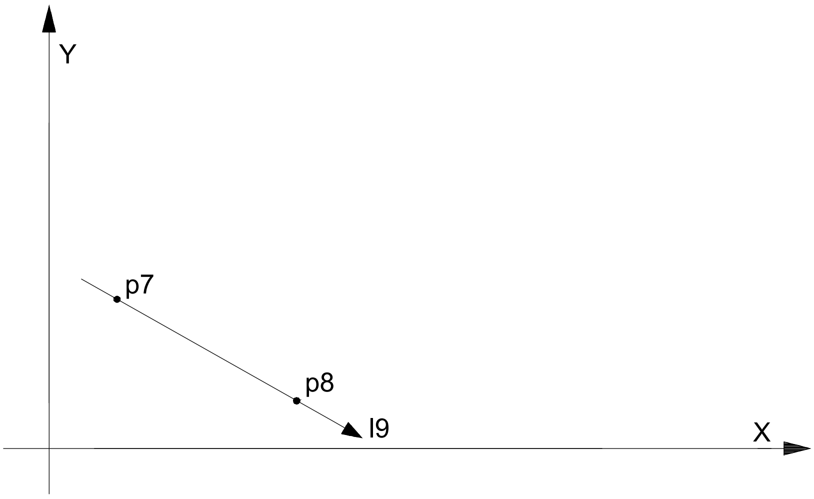
Straight line tangent to two circles

I3 = I25 J35 r-17,I70 J20 r13



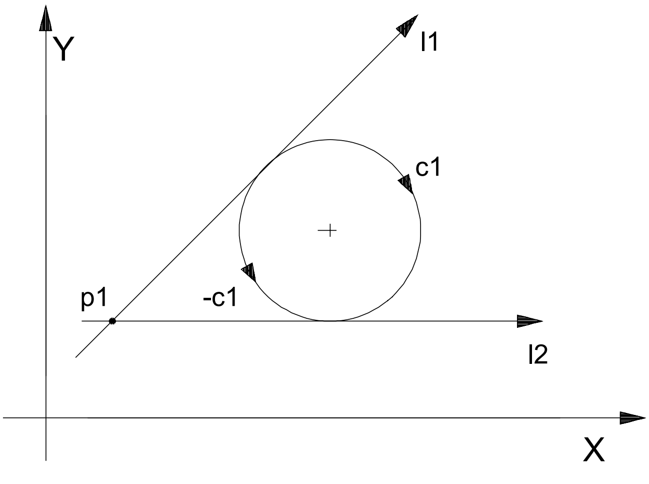
Straight line tangent to two circles

I4 = I25 J35 r17, I70 J20 r13



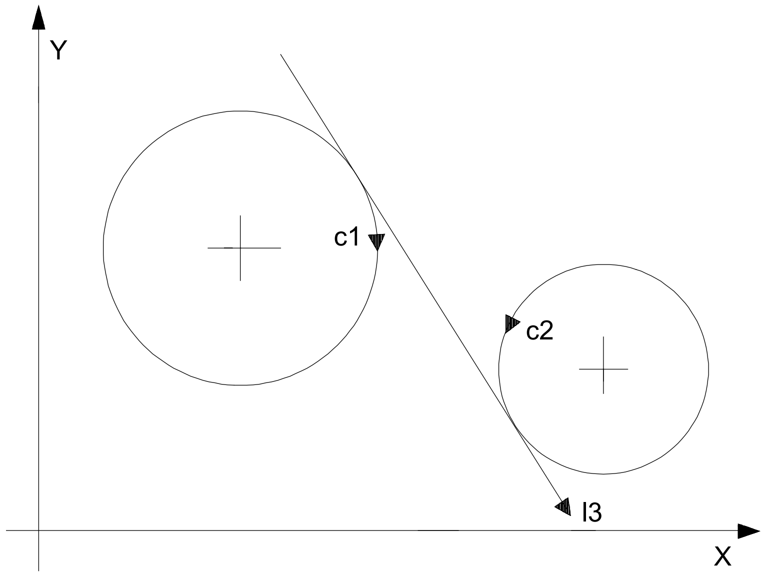
Straight line through two points

l9 = p7, p8



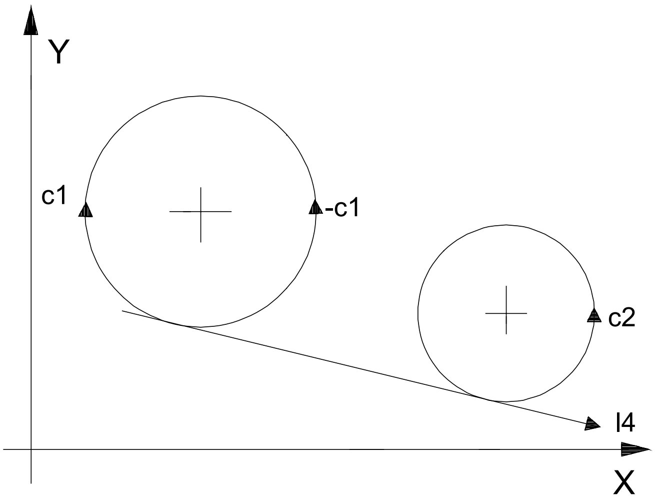
Straight line tangent to a circle and passing through a point

l1 = p1, c1
l2 = p1,-c1



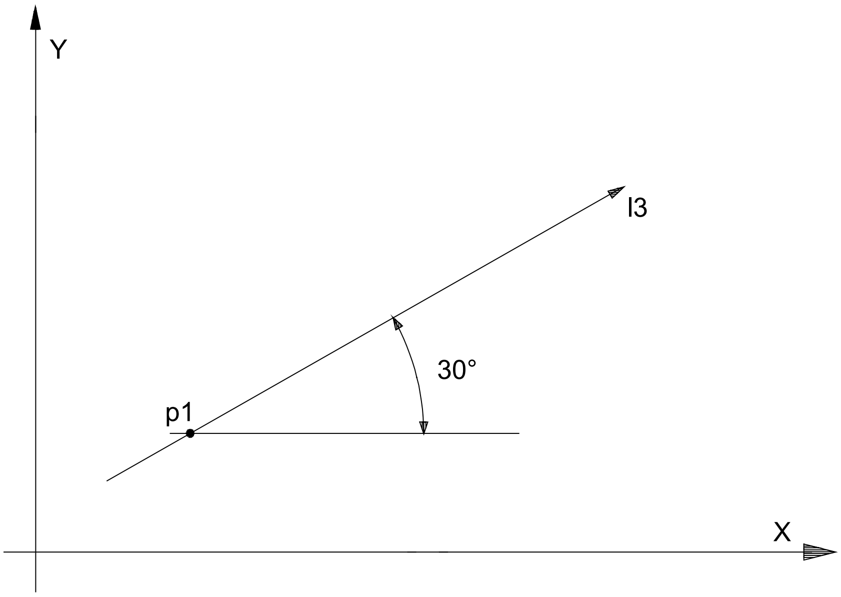
Straight line tangent to two circles

l3 = c1,c2



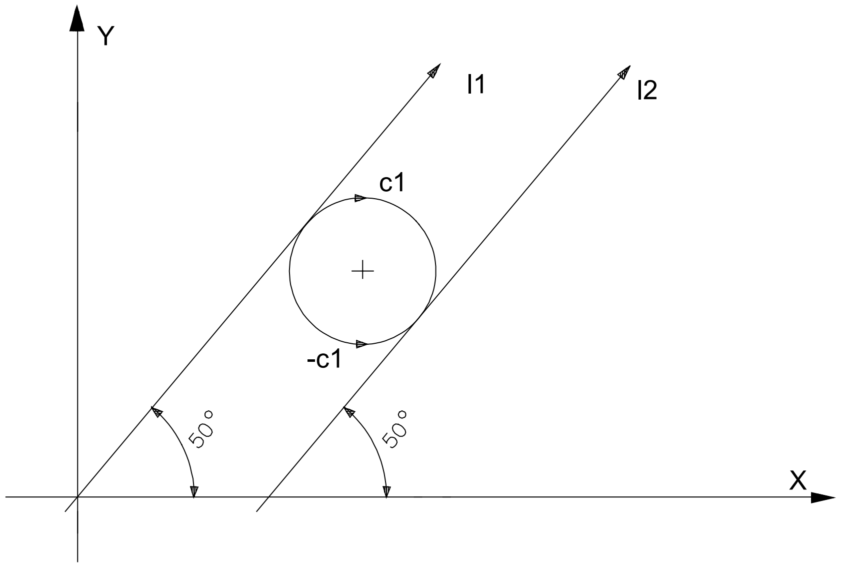
Straight line tangent to two circles

l4 = -c1,c2



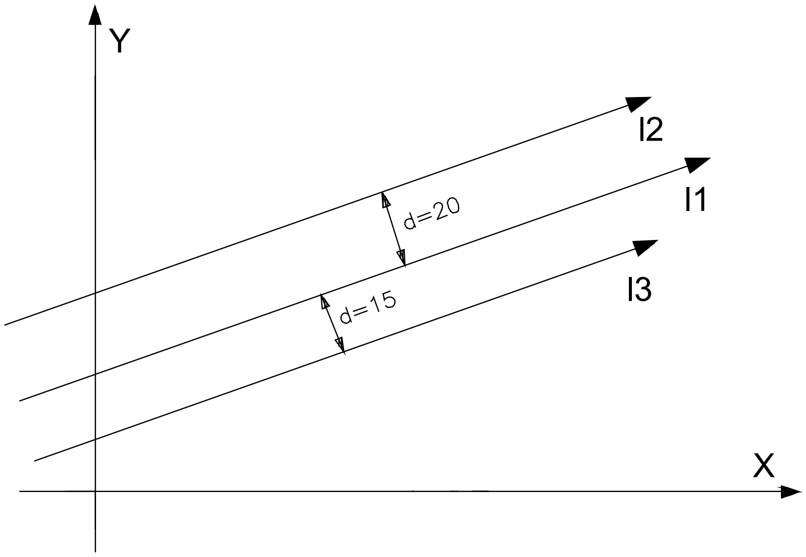
Straight line through a point and forming an angle with the abscissa axis

l3 = p1,a30



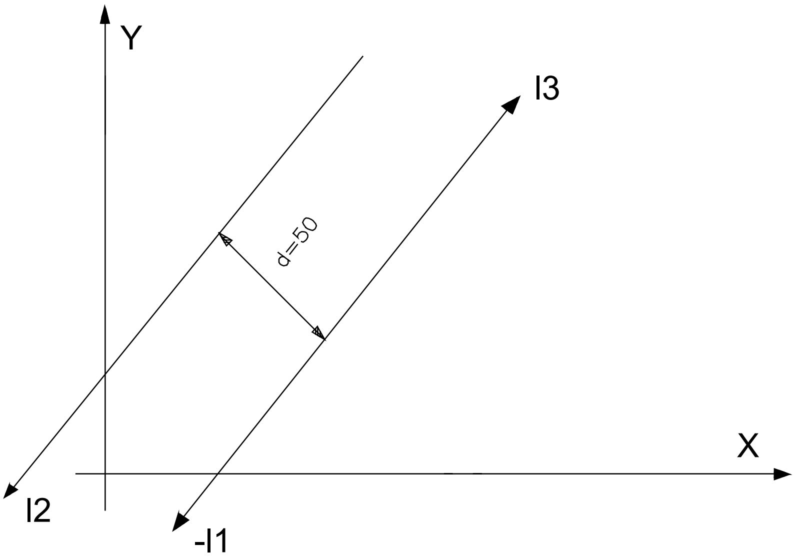
Straight line tangent to a circle and forming an angle with the abscissa axis

l1 = c1,a50
l2 = -c1,a50



Straight line parallel to
another straight lines

l2 = l1,d20
l3 = l1,d-15



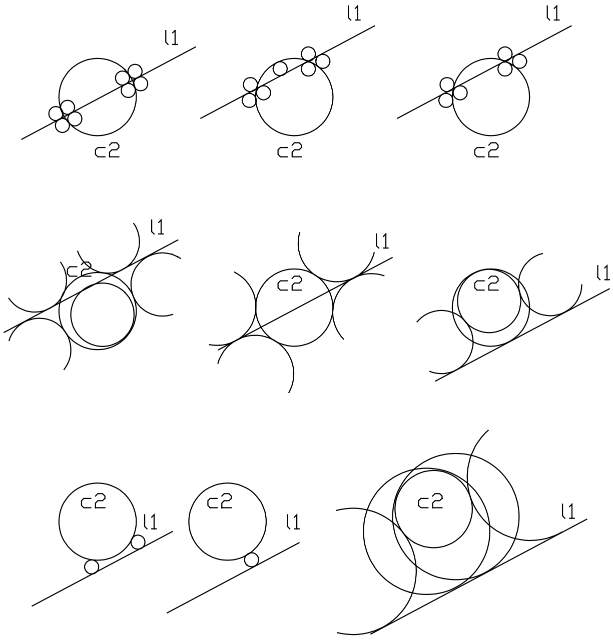
Straight line parallel to
another straight line

l2 = -l1,d-50

DEFINITION OF CIRCLES

Faction GTL permits to define circles both directly (explicitly) or indirectly (implicitly).

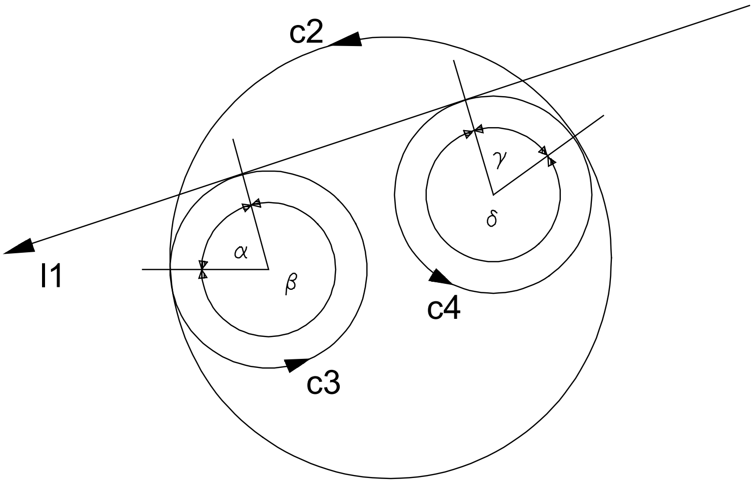
Description When defining circles indirectly, the programmer must check the compatibility between the directions of the circles and write a "-" sign to alter the direction of a predefined element. Given a circle of known radius and a straight line (and regardless of their directions), there are from 1 to 8 solutions to a circle tangent to two elements, as shown in the figure below:



Circles tangent to a straight line and a circle

After checking that the directions of the predefined elements are compatible with that of the circle to be defined, there are two possible solutions.

To select one of the two possible circles (both of which have the same radius and the same direction), it is necessary to check the direction of the elements specified in the definition and the two arcs in which the resulting circle will be divided by the points of tangency with the predefined elements. GTL always generates a circle that goes from the first to the second arc corresponding to the smallest central angle, as shown in the figure below:

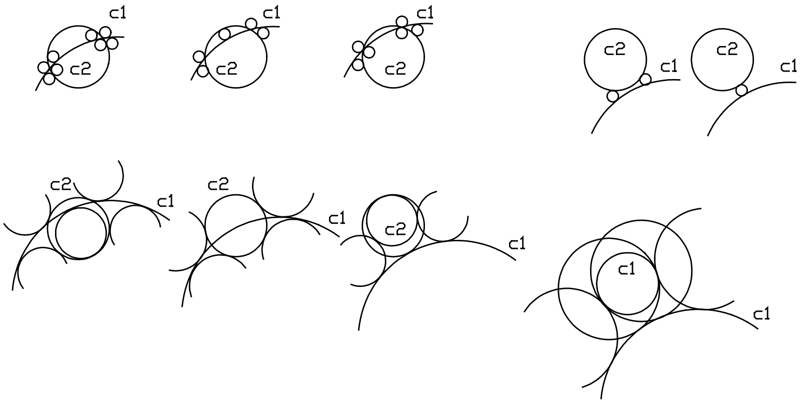


Tangent circles with the smallest ceangle

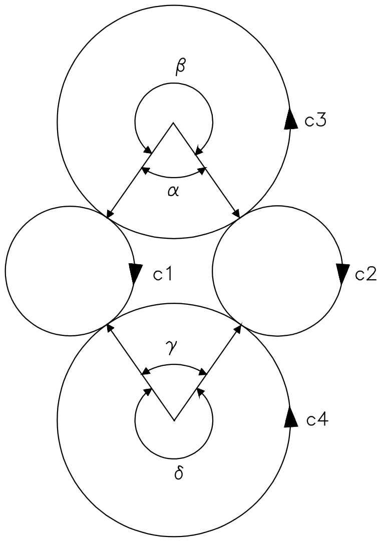
The c3 circle can be obtained by writing in the definition first the l1 straight line and then the c2 circle, because c3 allows to move from l1 to c2 (whose arc corresponds to the smallest angle).

The c4 circle can be obtained by writing in the definition first the c2 circle and then the l1 straight line, because c4 allows to move from the c2 (whose arc corresponds to the smallest angle) to l1.

The same criteria must be applied when programming a circle tangent to two predefined circles. Also in this case there are from 1 to 8 possible solutions, as shown in the figure below.



Circles tangent to two predefined circles



Circles tangent to two predefined circles

After checking compatibility of directions between predefined circles and programmed circles, there are only two possible solutions left.

To select one of them, consider the two arcs into which the new circle is divided.

To obtain c3, program first c1 and then c2.

To obtain c4, program first c2 and then c1.

Direct format

Circle given the cartesian coordinates of center and radius:

cn = [on] I.. J.. r..

Circle given the polar coordinates of center and radius:

cn = [on] m.. a.. r..

Indirect

Circle of known radius tangent to two predefined straight lines:

cn = [-] ln, ln, r..

Circle tangent to a straight line and predefined circle of given radius:

cn = [-]ln,[-]cn,r..

cn = [-]cn,[-]ln,r..

Circle of known radius through a predefined point, tangent to a predefined straight line:

cn = pn,[-]ln,r..

cn = [-]ln,pn,r

Circle of known radius tangent to a two predefined circles:

cn = [-]cn,[-]cn',r

Circle of known radius through a predefined point, tangent to a predefined circle:

cn = pn,[-]cn,r..

cn = [-]cn,pn,r..

Circle of known radius through two predefined points:

cn = pn,pn',r..

Circle whose center is a predefined point tangent to a predefined straight line:

cn = pn, [-]ln'

Circle whose center is a predefined point tangent to a predefined circle:

cn = pn,[-]cn [,s2]

Circle through three points:

cn = pn,pn',pn"

Circle of known radius whose center is a point:

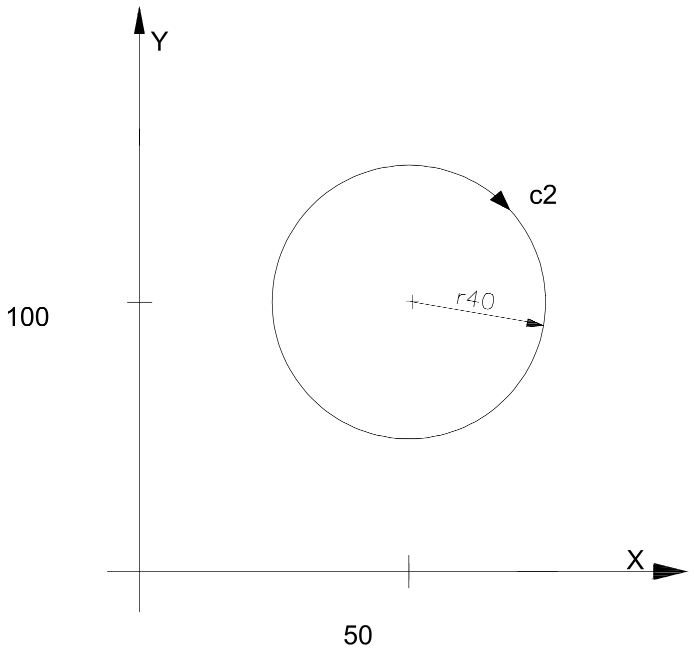
cn = pn,r..

Circle concentric to a predefined circle and lying at a known distance from it:

cn = [-]cn,d..

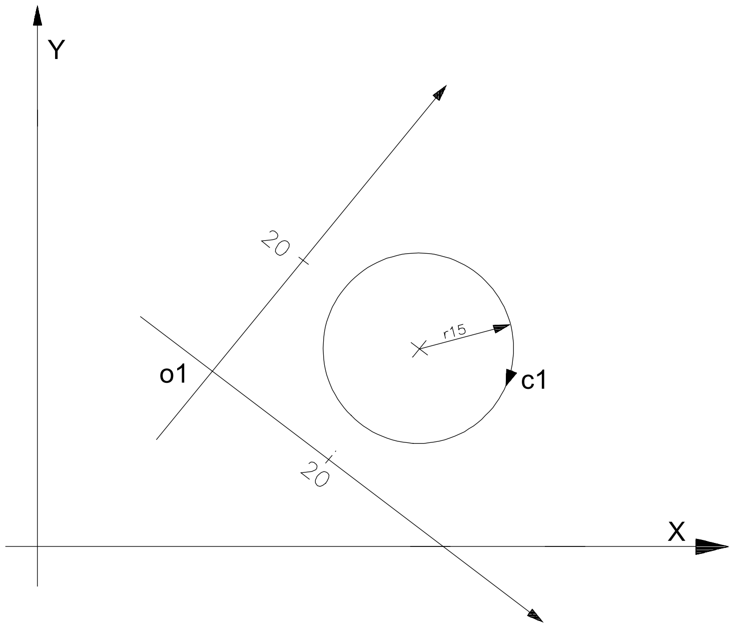
where

cn	is a circle whose n index is a number ranging from 0 to the configured maximum.
I..J..	circle center coordinates: I=abscissa, J=ordinate.
r..	circle radius (positive if counterclockwise, negative if clockwise).
[-]ln **[-]ln'**	predefined straight lines whose indexes are n and n'. To program the opposite direction write a "-" sign.
pn pn' **pn"**	predefined points with indexes n, n' ed n".
[-]cn **[-]cn'**	predefined circles with indexes n and n'. To program the opposite direction write a "-" sign.
[s2]	selects the largest possible circle.
d..	distance between two circles. Look in the direction of [-]cn: if cn' is on the right hand side, d is positive; cn' is on the left hand side, d is negative.



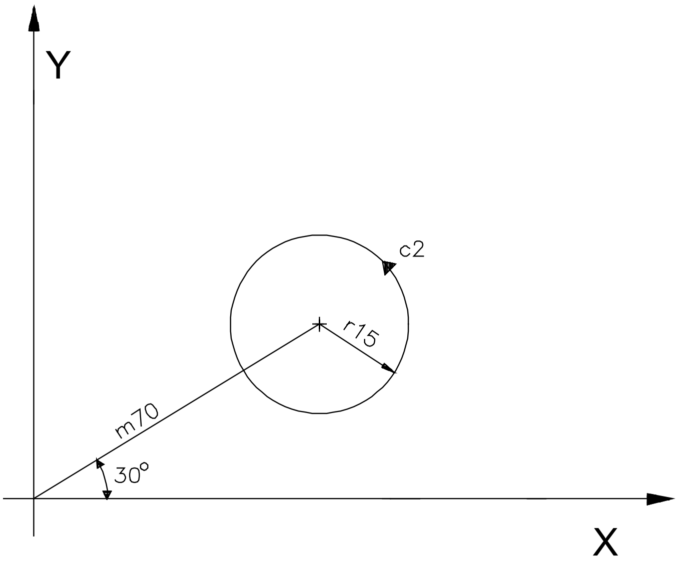
Circle with center and radius in cartesian coordinates

c2=I50J100r-40



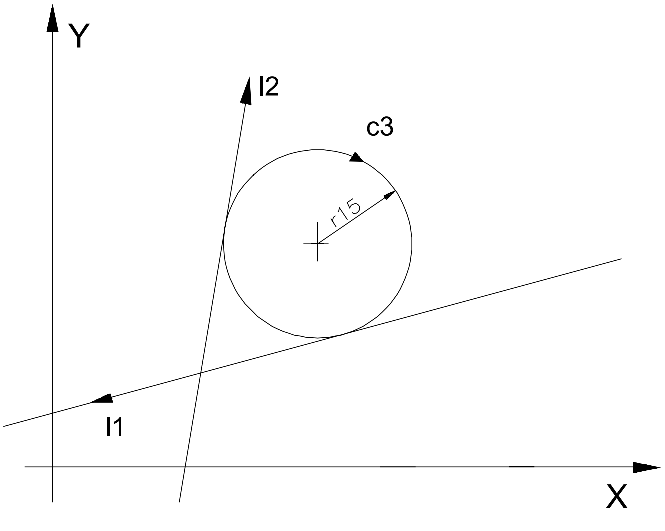
Circle with center and radius in cartesian coordinates

c1=o1I20J20r-15



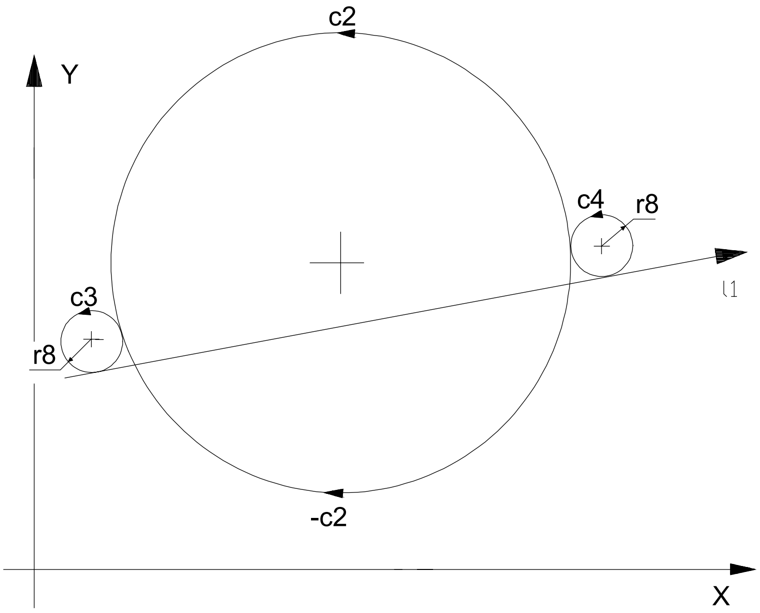
Circle with center and radius in polar coordinates

c2 = m70 a30 r15



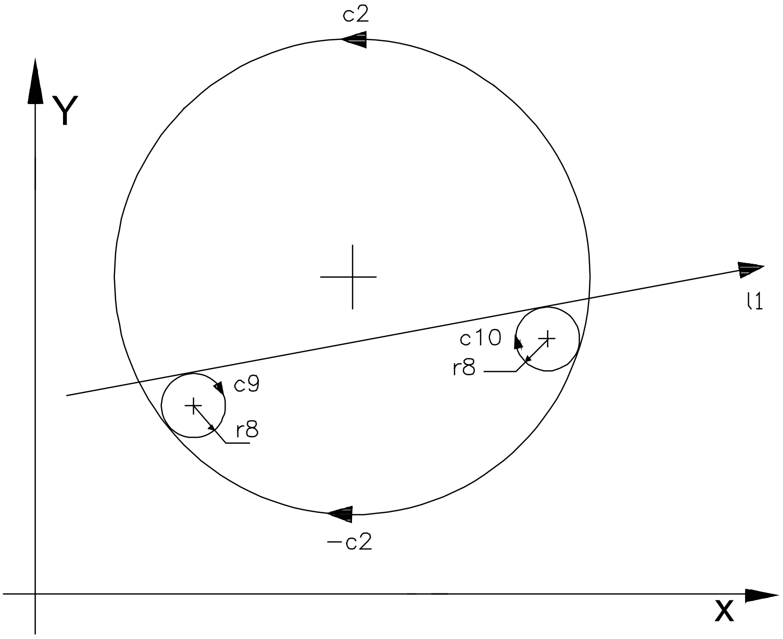
Circle of r radius tangent to two predefined straight lines

c3 = l1,l2,r-15



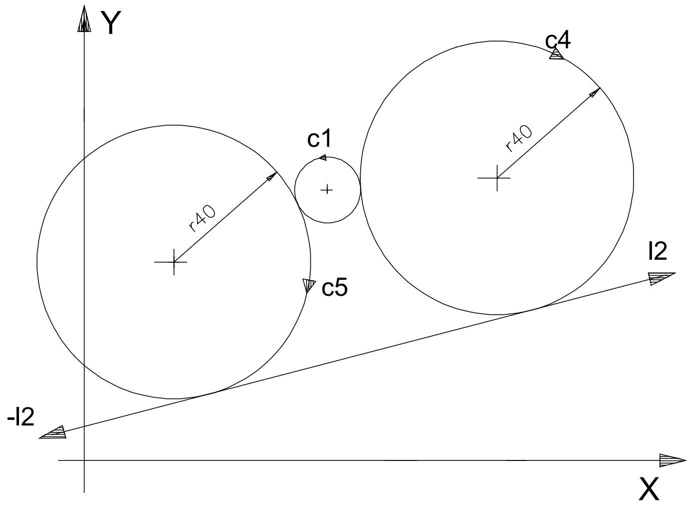
Circles tangent to a straight line and a circle

c3 =l1,-c2,r8
c4 = -c2,l1,r8



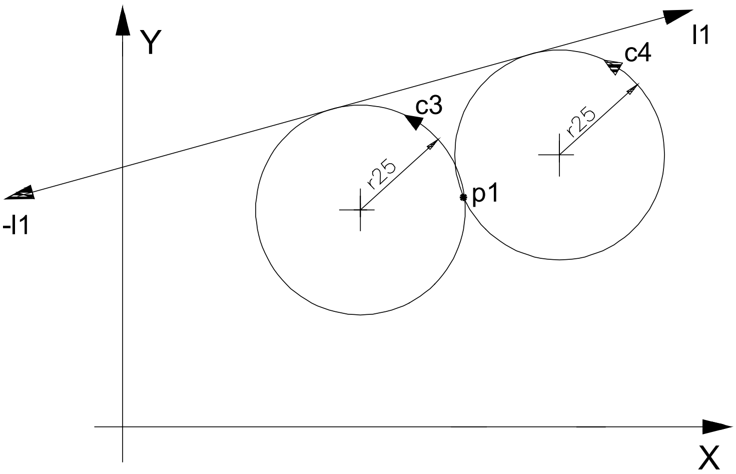
Circles tangent to a straight line and a circle

c9 = -c2,l1,r-8
c10 = l1,-c2,r-8



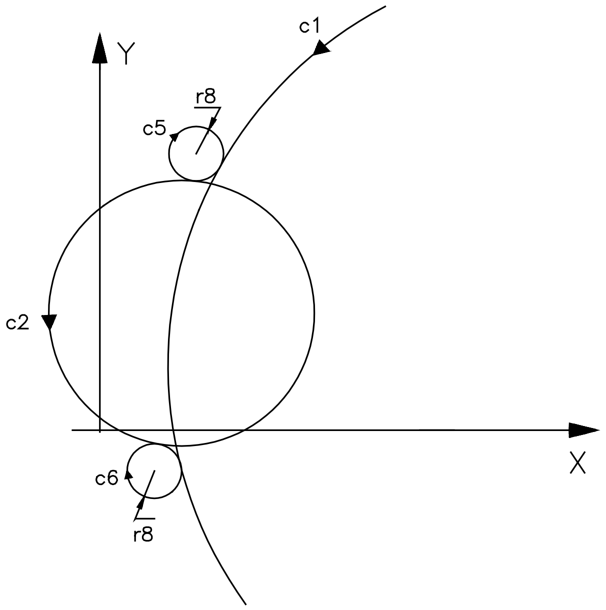
Circles tangent to a straight line and a circle

c4 = -l2,c1,r-40
c5 = c1,-l2,r-40



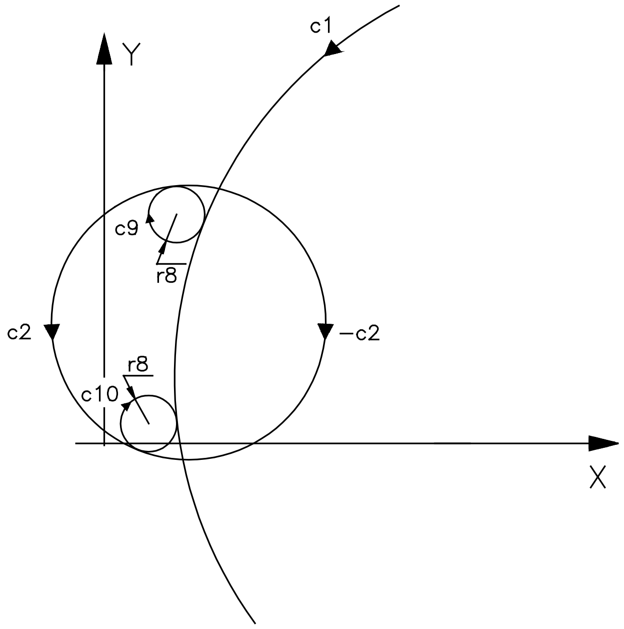
Circles through a point and tangent to a straight line

c3 = p1,-l1,r25
c4 = -l1,p1,r25



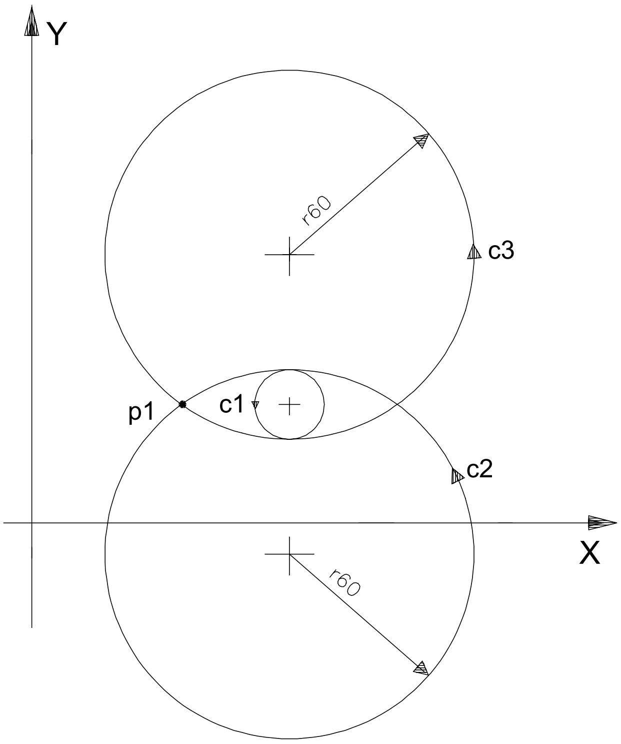
Circle tangent to two circles

c5 = c1,c2,r-8
c6 = c2,c1,r-8



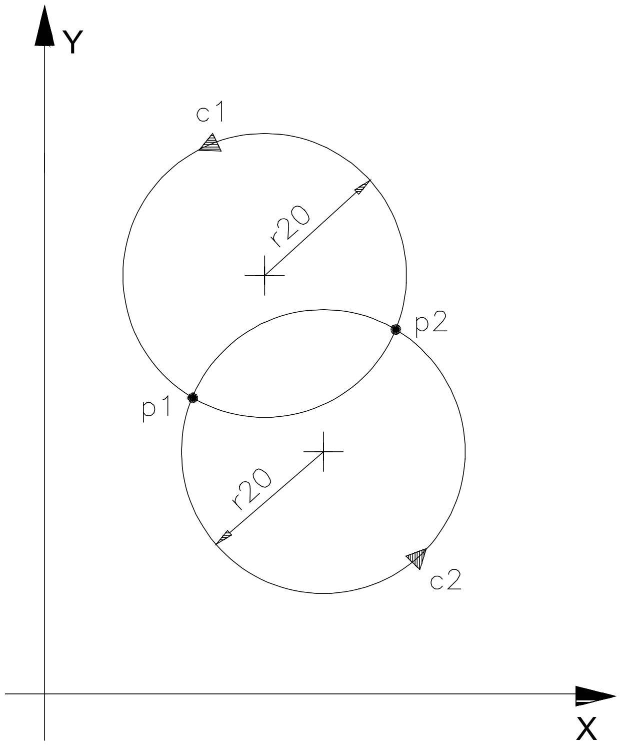
Circle tangent to two circles

c9 = -c2,c1,r-8
c10 = c1,-c2,r-8



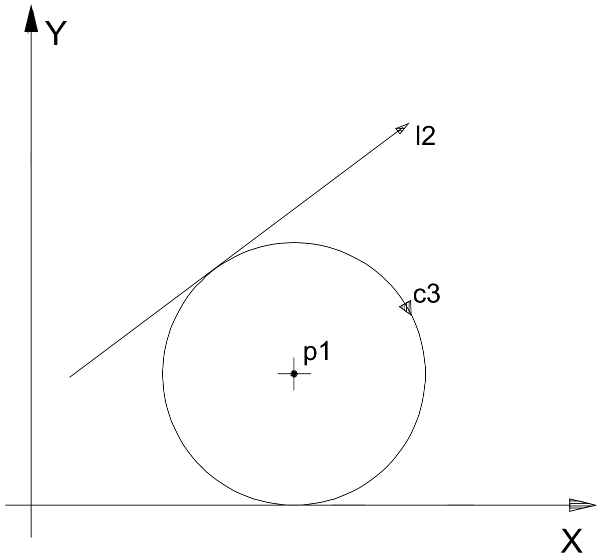
Circle through a point and tangent to a circle

c2 = c1,p1,r60
c3 = p1,c1,r60



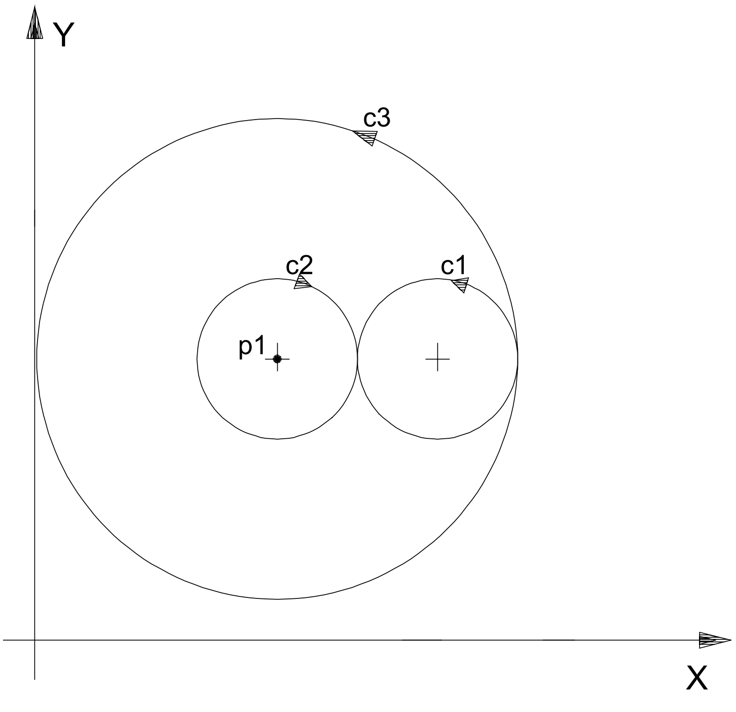
Circles through two points

c1 = p1,p2,r20
c2 = p2,p1,r20



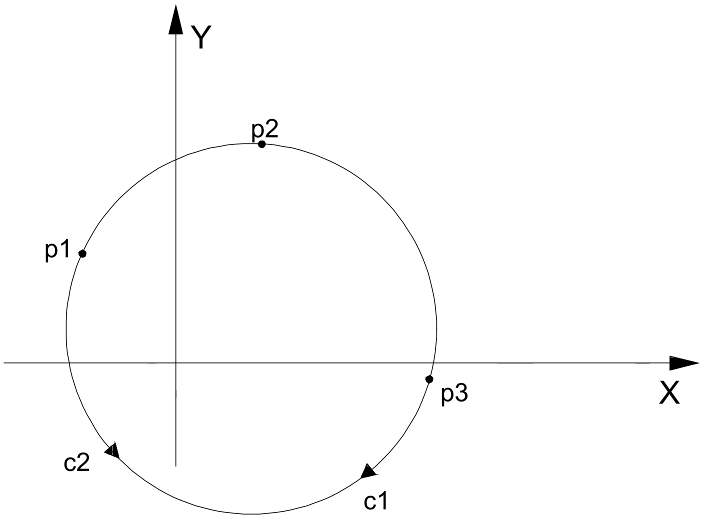
Circle with predefined center tangent to a straight line

c3 = p1,l2



Circle with predefined center tangent to another circle

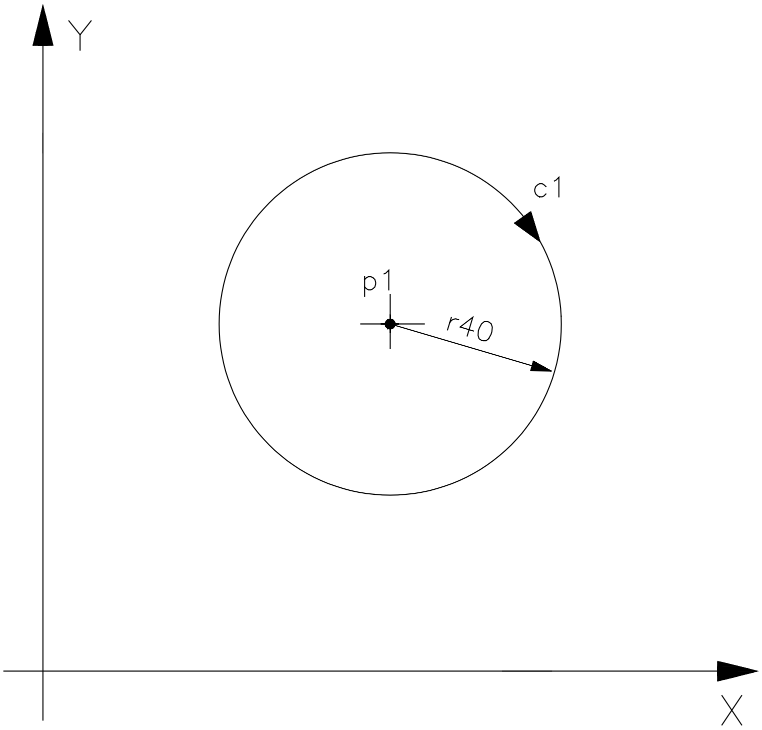
c2 = p1,c1
c3 = p1,c1,s2



Cerchio passante per tre punti

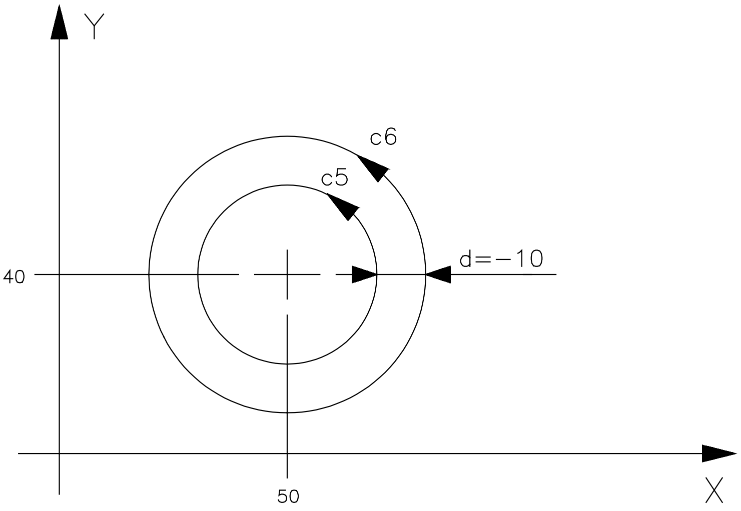
c1 = p1,p2,p3
c2 = p3,p2,p1

The circle goes from the first to the second point and then to the third point.



Circle whose center is a point

c1 = p1,r-40



Circle concentric to another circle

c6 = c5,d-10

DEFINITION OF A PROFILE

Profile start and end

A profile programmed in GTL is delimited by two codes:

G21 identifies the profile start

G20 identifies the profile end

Profile types

A profile is a sequence of predefined geometric elements. Profiles may be open or closed.

Open profiles

In an open profile the starting point is different from the end point (pn).
Tool radius offset is perpendicular to the first point of the first element and to the last point of the last element.
To open tool offset radius on the first profile point you must program G21 G41/G42 in the block. To close tool offset radius on the last profile point you must program G20 G40 in the block.

Closed profiles

To obtain a closed profile you must first program the last element and then re-call the first programmed element.
The first offset point in the profile is the intersection between the first and the last translated elements (first point = last point).
Tool radius offset must be opened at profile start, i.e. on the block that programms the last element, by programming codes G21 G41/G42. To close tool offset radius at profile end you must program G21 G41/G42 in the block.
If the first or the last element are circles, there are two possible intersections. By default the system selects the first intersection. To select the second intersection, write s2 both in the block that programs the last element at profile start and in the block that programs the last element at profile end.

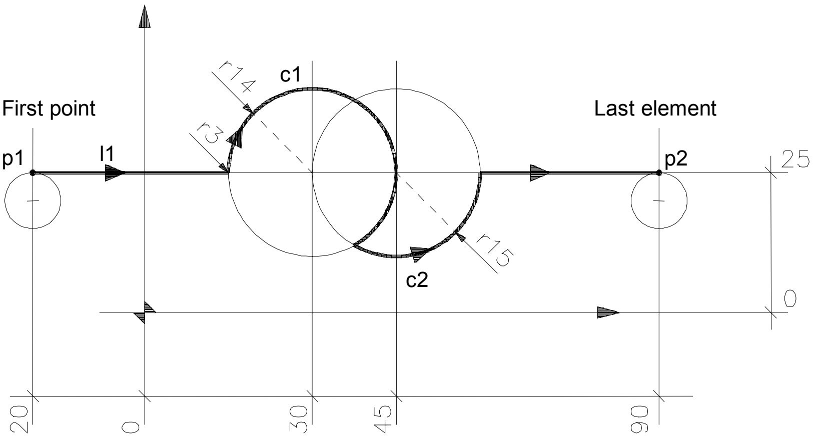
Allowed G codes

The only G codes accepeted by GTL are as follows: G27, G28, G04, G09, G41 and G42.

IMPORTANT

If pn is programmed in the starting profile block then the profile is open. pn may be programmed both at profile start and at profile end but cannot be programmed in the profile.

Open profile



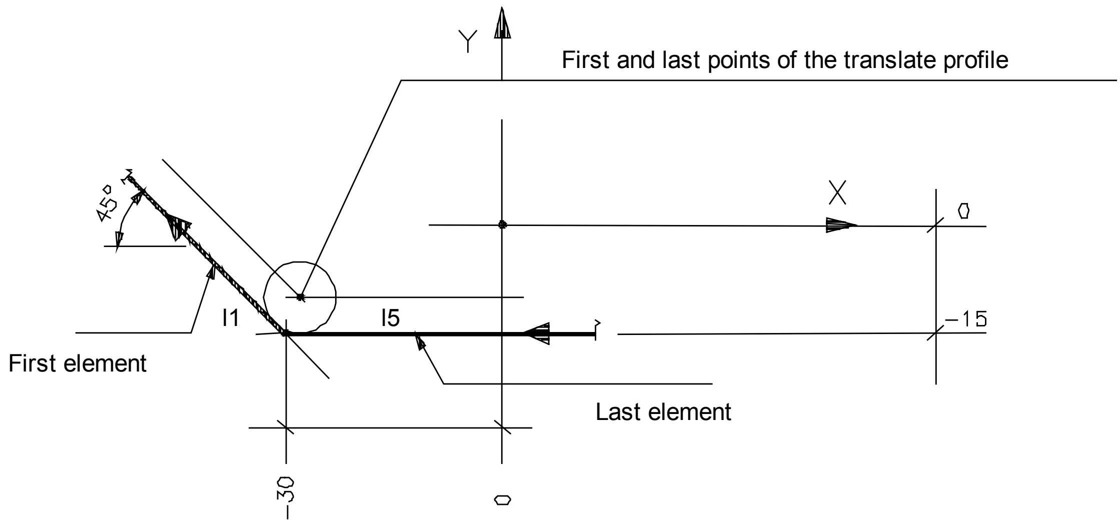
Example of open profile

```
................
l1       = X Y25,a
p1       = X-20 Y25
p2       = X90 Y25
c1       = I30 J25 r-14
c2       = I45 J25 r15
................
G21 G42 p1            - first point
l1
r3
c1 s2
c2 s2
l1
G20 G40 p2            - last point
................
```

IMPORTANT

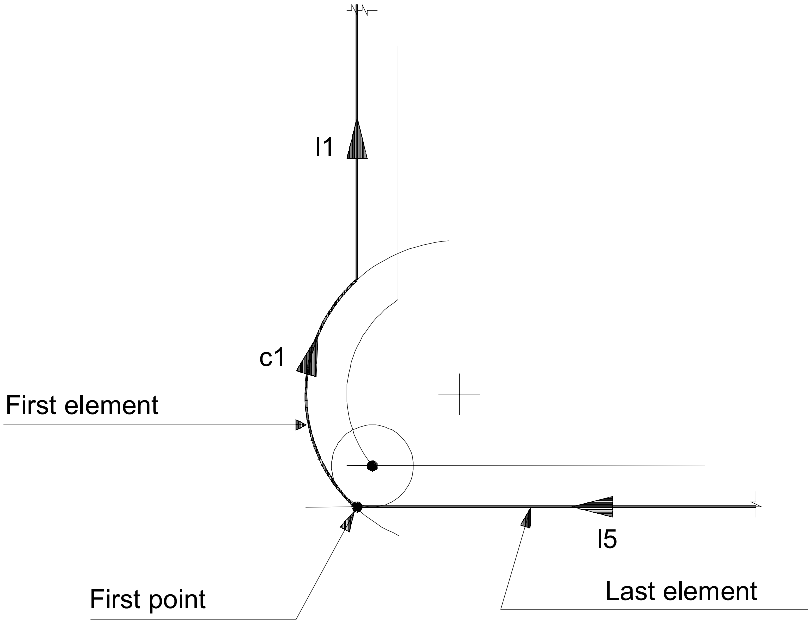
Tool radius offset must start on the first point of the profile and end on the last point. Tool radius offset is disabled on the first motion block of the plane axis programmed after G40.

Closed profile



Example of straight line-straight line closed profile

```
.................
I5       = X Y-15,a180
.................
I1       = X-30 Y-15,a135
.................
G21 G42 I5             - last element
I1                     - first element
.................
I5                     - last element
G20 G40 I1             - first element
```



Example of straight line-circle closed profile

```
c1        = I.. J.. r
....................
l1        = X.. Y.. a90
....................
l5        = X.. Y..,a180
....................
G21 G42 l5 s2             - last element
c1 s2                     - first element
l1
....................
l5 s2                     - last element
G40 c1                    - first element
....................
```

IMPORTANT

Tool radius offset must be opened on the block that programs the last element of the profile and end on the block that programs the first element of the profile. Tool radius offset is disabled on the first motion block of the plane axis programmed after G40.

Spindle axis motion

Non-contouring axes can be moved on any point in the profile, for example, to penetrate the part.

Spindle axis motion at the first point must be programmed after the point in open profiles, and between the definition of the last support element and that of the first element in closed profiles.

Examples:

```
. . . . . . . .
G21G42 p1
Z-10
l1
. . . . . . .
```

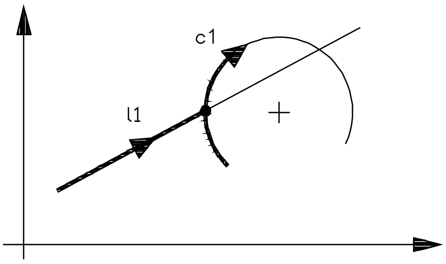
```
. . . . . . . .
G21G42 l5
Z-10
l1
. . . . . . .
```

Connecting the elements

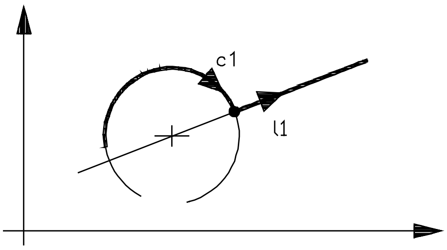
The geometric elements of the profile may be interlinked by tangential or intersecting lines, by an automatic radius or by a bevel.

1. Intersection between elements

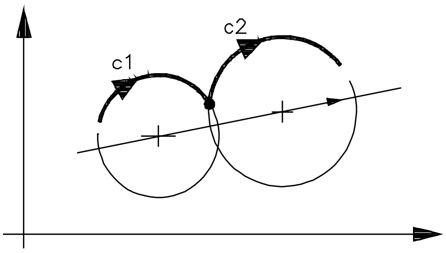
The intersection between two straight lines has only one solution.
In straight line-circle intersections the default solution is the first of the two possible intersections. To select the second intersection it is necessary to program the s2 discriminator after the definition of the first element.



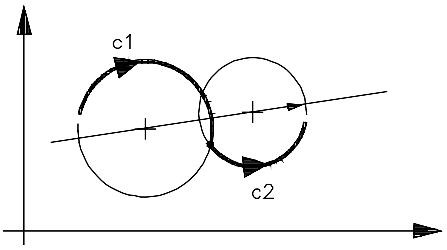
```
N24 l1
N25 c1
```



```
N28 c1 s2
N29 l1
```



```
N33 c1
N34 c2
```

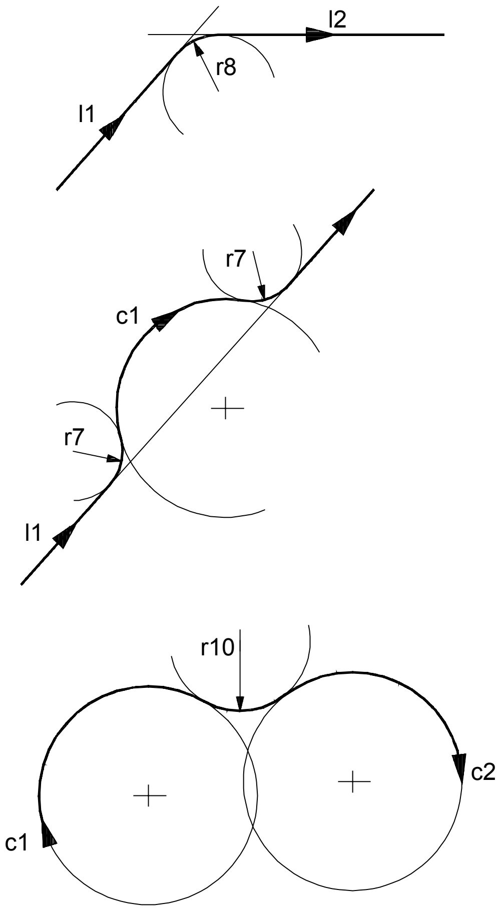


```
N36 c1 s2
N37 c2
```

In straight line-circle intersections, the sequence of solutions is given by the direction of the line. In circle-circle intersections the first intersection is on the left hand side of the straight line that links the centers of the first and second circle, whereas the second intersection is on the left hand side of the straight line that links the centers of the first and second circle.

2. Automatic radiuses

GTL permits to define radiuses between elements by simply programming the radius length and sign. A positive value defines a counterclockwise radius; a negative value defines a clockwise radius.



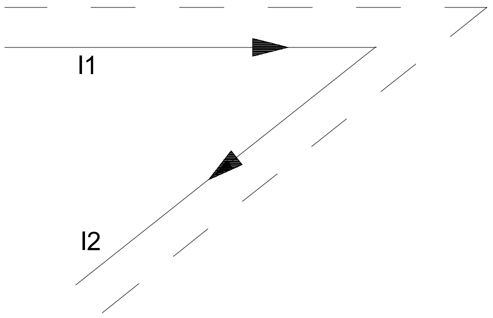
```
...................
N20 l1
N21 r-8
N22 l2
...................
```

```
...................
N24 l1
N25 r7
N26 c1
N27 r7
N28 l1
```

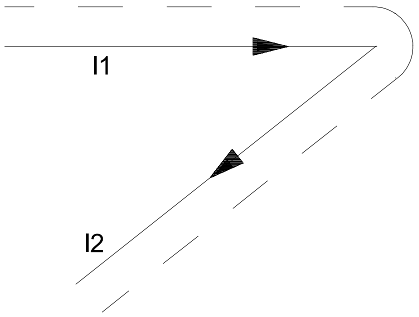
```
...................
N90 c1
N91 r10
N92 c2
...................
```

Automatic radius between elements

If tool offset is enabled, the tool is positioned on the intersection between both geometric elements translated by the tool radius. To insert a radius between the elements you must program an r0 radius.



```
no radius
.........................
l1
l2
.........................
```



```
radius r0
.........................
l1
r0
l2
.........................
```

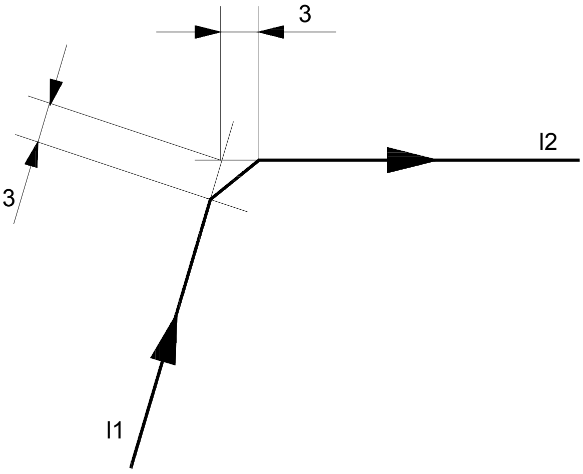
Zero radius with intersection and tool offset

IMPORTANT

The block with the r radius cannot be programmed either inmmediately after the block that includes the G21 or immediately before the block that includes the G20 code. In other words, the profile cannot either start or end with a radius.

3. Bevels

GTL permits to define a bevel between two lines by programming the bevel length without sign. The bevel length is the distance to the intersection point.



```
N30 l1
N31 b3          Bevel
N32 l2
```

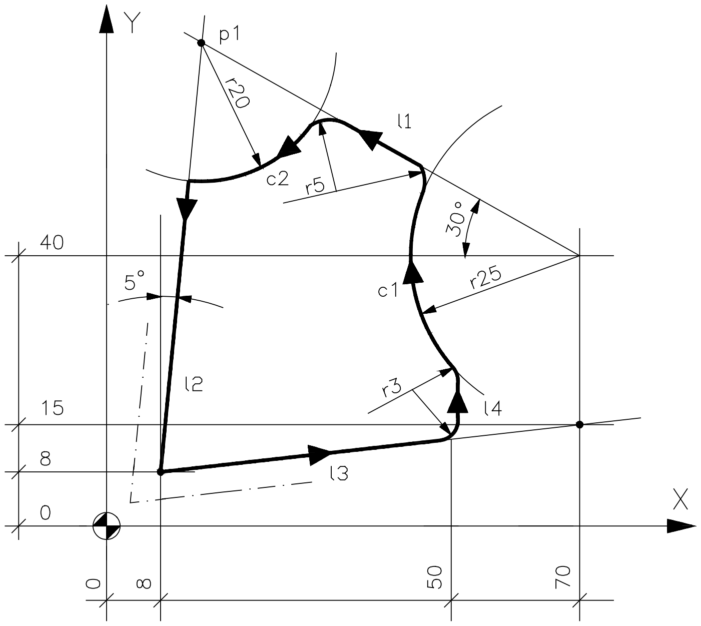
NOTE:

- The block with the b bevel cannot be programmed either inmmediately after the block that includes the G21 or immediately before the block that includes the G20 code. In other words, the profile cannot either start or end with a bevel.
- With GTL displacements occur always at the configured feedrate. To program a move at a rapid rate it is necessary to declare a very high F.
- If the interpolation plane is not formed by axes X and Y you must define the plane and then program the elements with respect to the its abscissa and ordinate.

Example:

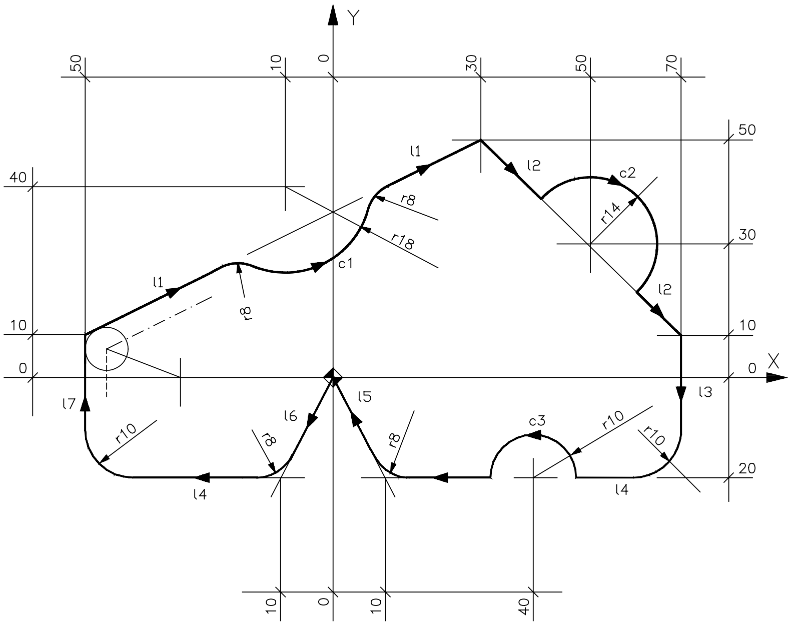
```
N1    G16BY
N2    l1 = B70,Y40,a150
N3    l2 = B8,Y8,a-95
N4    p1, = l1,l2
N5    l3 = B8,Y8,B70,Y15
N6    c1 = I70,J40,r-25
.........
.........
N12   G21 G42 l2
.........
```

EXAMPLES OF GTL PROGRAMMING



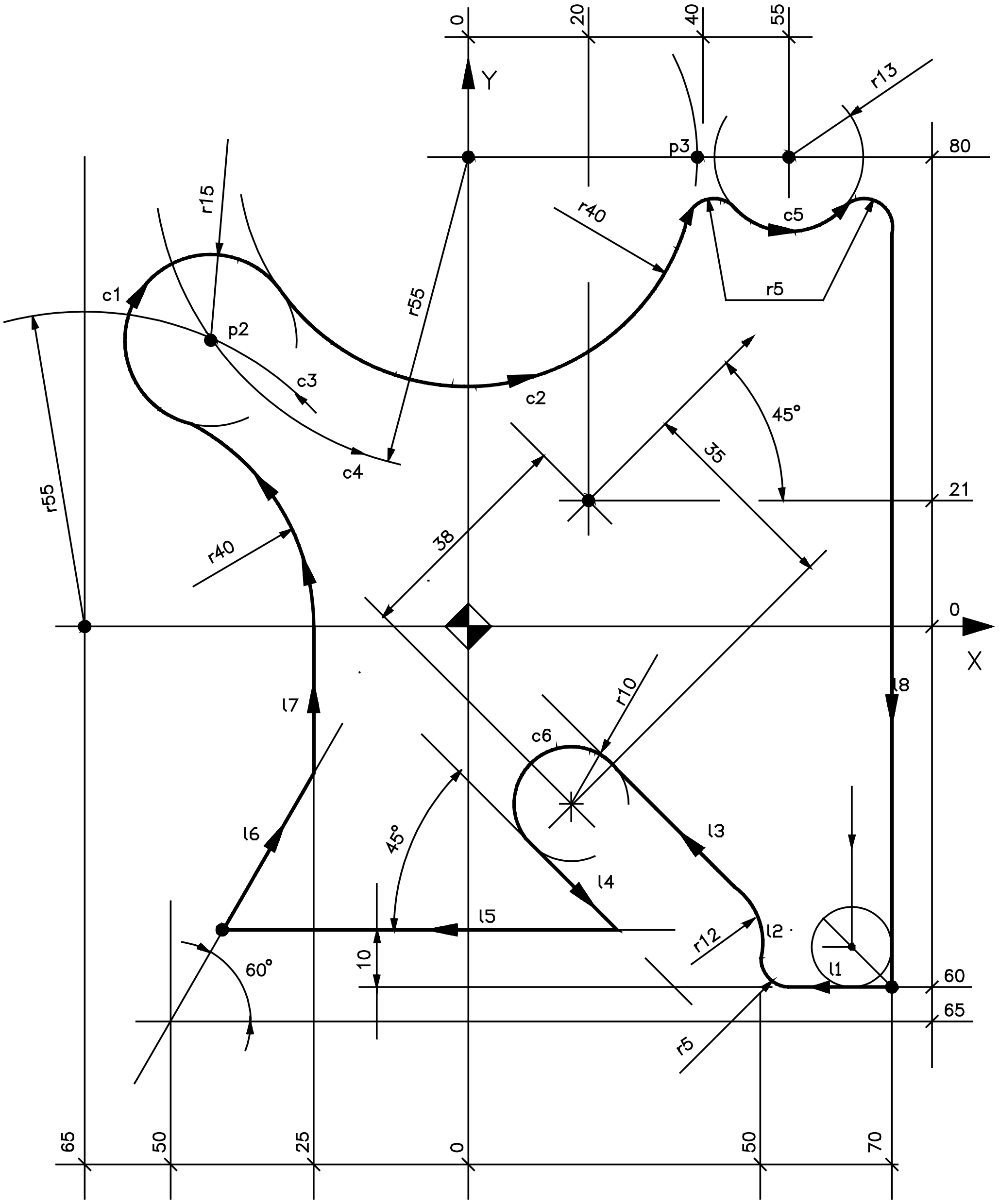
Example 1

```
N1   (DIS,"EXAMPLE GTL")
N2   l1=X70Y40,a150
N3   l2=X8Y8,a-95
N4   p1=l1,l2
N5   l3=X8Y8,X70Y15
N6   l4=X50Y,a90
N7   c1=I70J40 r-25
N8   c2=p1,r-20
N9   F250 S800 T1.1 M6M3
N10 GXY
N11 Z-10
N12 G21G42l2
N13 l3
N14 r3
N15 l4
N16 r3
N17 c1
N18 r5
N19 l1
N20 r5
N21 c2s2
N22 l2
N24 G20 G40 l3
N25 GZ2
N23 XY M30
```



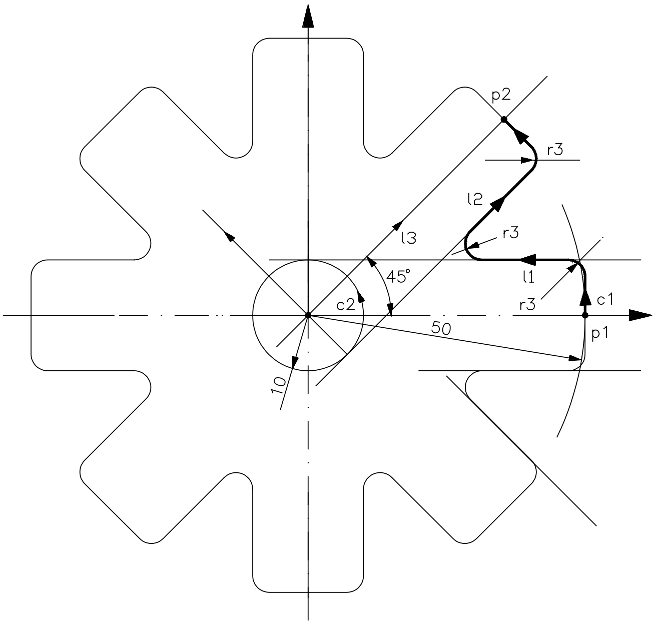
Example 2

```
N1    (DIS,"EXAMPLE GTL")
N2    l1=X-50Y10,X30Y50
N3    l2=X30Y50,X70Y10
N4    l3=X70Y0,a-90
N5    l4=X=Y-20,a180
N6    l5=X10Y-20,X0Y0
N7    l6=X0Y0,X-10Y-20
N8    l7=X-50Y0,a90
N9    c1=I-10J40r18
N10   c2=I50J30r-14
N11   c3=I40J-20r10
N12   S...M...T3.3M6M....
N13   G0X-30Y0
N14   Z-10
N15   G21G42l7
N16   l1
N17   r-8
N18   c1
N19   r-8
N20   l1
N21   l2
N22   c2s2
N23   l2
N24   l3
N25   r-10
N26   l4
N27   c3s2
N28   l4
N29   r-8
N30   l5
N31   l6
N32   r-8
N33   l4
N34   r -10
N35   l7
N36   G20G40l1
N37   G0Z20
```


0
20
40
55
Y
r13
p3
80
r40
c5
r15
r5
c1
r55
p2
c3
c2
45°
c4
35
21
r55
38
r40
0
X
r10
l8
l7
c6
l3
l6
45°
l4
l5
r12
l2
60°
10
l1
60
65
r5
65
50
25
0
50
70

Example 3

```
N1      (DIS,"EXAMPLE 3")
N2      S...F...T1.1M6 M...
N3      o1=X20 Y21 a45
N4      l1=X0 Y-60,a180
N5      l2=X50 Y0,a90
N6      c6=o1 I-38 J-35 r10
N7      l3=c6,a135
N8      l4=c6,a-45
N9      l5=X0 Y-50,a180
N10     l6=X-50 Y-65,a60
N11     l7=X-25 Y0,a90
N12     c3=I-65 J0 r55
N13     c4=I0 J80 r55
N14     p2=c3,c4
N15     c1=p2,r-15
N16     p3=X40 Y80
N17     c2=c1,p3,r40
N18     c5=I55 J80 r13
N19     l8=X70 Y0,a-90
N20     G21 G42 l8
N21     Z-10
N22     l1
N23     r-5
N24     l2
N25     r12
N26     l3
N27     c6
N28     l4
N29     l5
N30     l6
N31     l7
N32     r40
N33     c1
N34     c2
N35     r-5
N36     c5
N37     r-5
N38     l8
N39     G20 G40 l1
N40     G Z20
N41     X... Y... M30
```



Example 4: 8 profile repetitions executed with 2 passes

```
N1       (DIS,"GTL EXAMPLE WITH ROTATION")
N2       F...S...T2.2 M6
N3       MSA=2
N4       p1=X50 Y0
N5       c1=I0 J0 r50
N6       c2=I0 J0 r10
N7       l1=c2,a180
N8       l3=X0 Y0,a45
N9       l2=c2,a45
N10      p2=l3,c1,s2
N11      GX60 Y0
N12      Z-2
"START"  N13
N14      E25=0
N15      (RPT,8)
N16      (ROT,E25)
N17      G21 G42 p1
N18      c1
N19      r3
N20      l1
N21      r-3
N22      l2
N23      r3
N24      c1
N25      G20 G40 p2
N26      E25=E25+45
N27      (ERP)
N28      (ROT,0)
"END"    N29
N30      MSA=0
N31      (EPP,START,END)
N32      GZ20
N33      XY M30
```

END OF CHAPTER

Chapter **16**

WORKING CYCLES FOR TURNING SYSTEMS

PROFILE PROGRAMMING

The profile is a program composed by blocks of movement that can be recalled by a roughing or finishing cycle.

There are three methods to program the profiles:

1. Write the profile in ISO language using the line editor

```
G1Z0X30
Z-15
Z-20.80127X50.09618
G3 Z-2629165X61.07694 I-33.79165 J35.09618
G1 Z-55.82309 X95.17692
G3 Z-64.82309 X100 I-64.82309 J64
G1 Z-76
X140
```

2. Write the profile in GTL language using the line editor

```
p1=Z0 X40       l1
l1=p1, a180     r-5
c1=I-40 J40     c1
l2=c1, a180     l2
p2=Z-55 X100  r-5
l3=p2, a90      l3
G21 p1          G20 p2
```

3. Write the profile with the graphic editor.In this case the profile is formed with the support of the direct geometry available at the editor level and therefore the program will be translated in output into the ISO language.

For further information , see the User Manual.

Restrictions to the definition of a profile to be recalled by the macro-instructions of roughing/finishing.

- The profile must be completely described in a subroutine, indicated in the following pages as “profile_name”.
- The blocks contained in the subroutine will have to be ISO/GTL types only, and only the programming of :
  - Axis names and co-ordinates
  - Movement G functions (G1, G2, G3 and possible operands)
  - Therefore, in general the three-letter commands will NOT be taken into consideration and neither will any commands that might modify the profile description (origins, mirror, scale factor)

NOTE:

The non-conformity to the listed points, might lead to a wrong interpretation of the profile described in the subroutine.

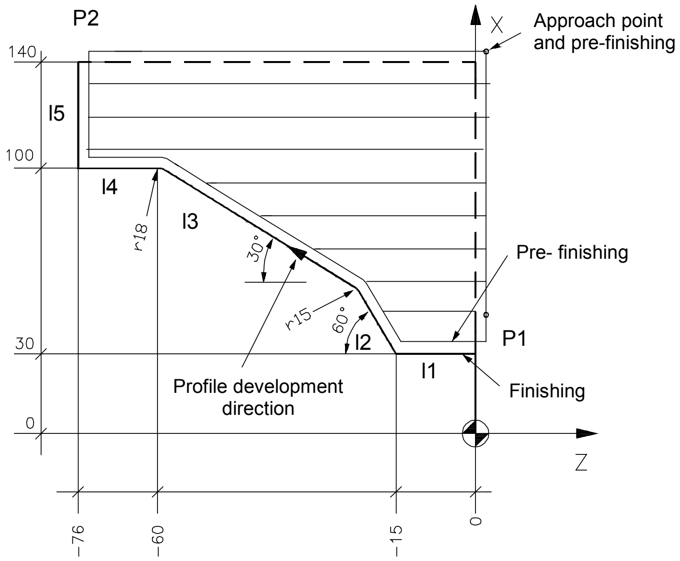


Figure 16-1 Para-axial roughing with pre-finishing and finishing

```
T1.1 M6 S.. F..
G X143 Z1.5
(SPF,PROF2,Z,9,X1,Z1)
X300 Z200
T2.2 M6 S.. F..
G X30 Z2
(CLP,PROF2)
G X300 Z200
..........
```

PROF2

```
p1 = Z0X30
l1 = p1, a180
l2 = Z-15X30, a120
l3 = Z-60X100, a150
l4 = Z-75X100, a180
p2 = Z-75X140
l5 = p2, a90
G21 p1
l1
l2
r15
l3
r18
l4
l5
G20 p2
```

SPECIAL CYCLES PROGRAMMING

The following three-letter codes allow defining and recalling of special cycles inside the program:

SPA	para-axial roughing without pre-finishing
SPF	para-axial roughing with pre-finishing
SPP	roughing parallel to the profile
CLP	finishing cycle
FIL	threading cycle
TGL	groove cutting cycle

MACRO-INSTRUCTIONS OF PARA-AXIAL ROUGHING WITHOUT PRE-FINISHING

Profile roughing, with cutting passes parallel to the ordinate or abscissa axis.

(SPA,profile_name, axis_name, n_pass. [,axis1 stock] [,axis2 stock])

Example : (SPA,PROF1,Z,6,X1,Z2)

where:

profile_name	is an ASCII string representing the part program name in which the profile is described.
axis_name	specifies the abscissa or ordinate name (Z o X) along which the roughing passes must be executed or in which the programmed stock has to be left.
n_pass	specifies the number of roughing passes . It may be a value between 1 and 255.
axis1 stock	indicates the machining allowance to be left on axis 1 (normally the X axis). The value must be positive.
axis2 stock	indicates the machining allowance to be left on axis 2 (normally the Z axis). The value must always be positive.

The stock values must be always positive.

The control decides automatically if the roughing refers to an internal or an external form by taking into consideration the approach point and the profile direction, and therefore assigns the correct sign to stock allowance.

The approach point must be external to the roughing field by at least as much as the programmed stock allowance.

If the profile is not monotonous, (that is it contains cavities), during the executing of the roughing cycle these are automatically passed over.

IMPORTANT

The profile must be monotonous for the roughing axis ,otherwise the system will display an error.
Therefore, the profile may contains cavities, but as regards the points trend along the roughing axis, they must be always increasing or decreasing.

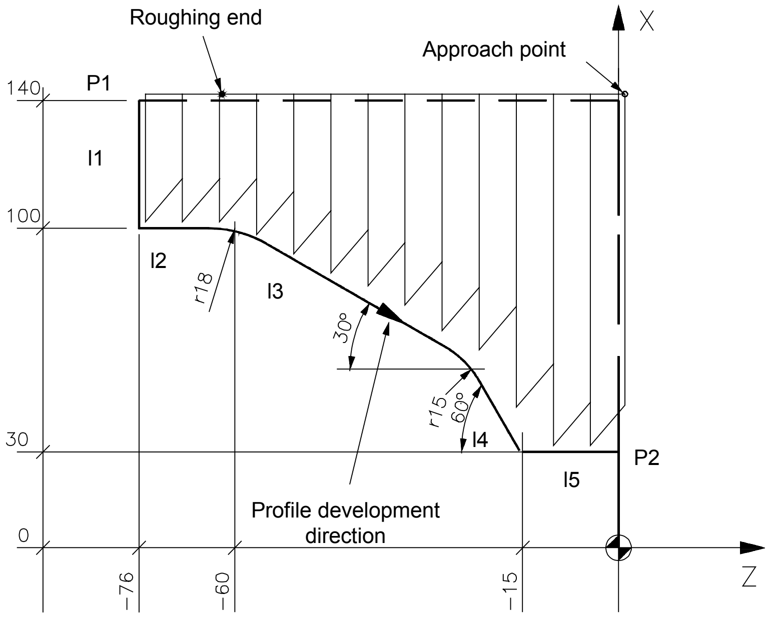


Figure 16-2 Example of a profile roughing with working passes parallel to the X axis

```
..........
..........
T1.1 M6 S.. F..
G X143 Z1.5
(SPA,PROF1,X,12,X1,Z1)
..........
..........
```

```
PROF1
p1 = Z-76X140
l1 = p1, a-90
l2 = Z0X100, a0
l3 = Z-60X100, a-30
l4 = Z-15X30, a-60
p2 = Z0X30
l5 = p2, a0
G21 p1
l1
l2
r-18
l3
r-15
l4
l5
G20 p2
```

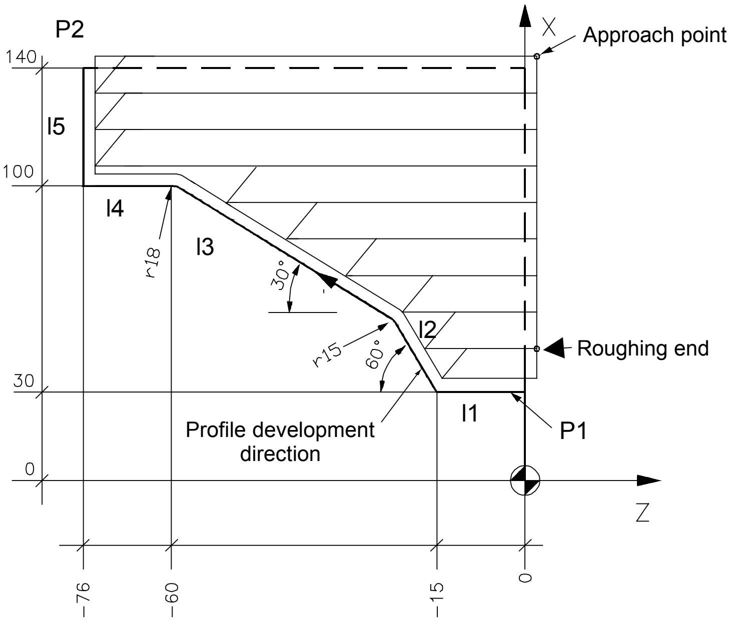
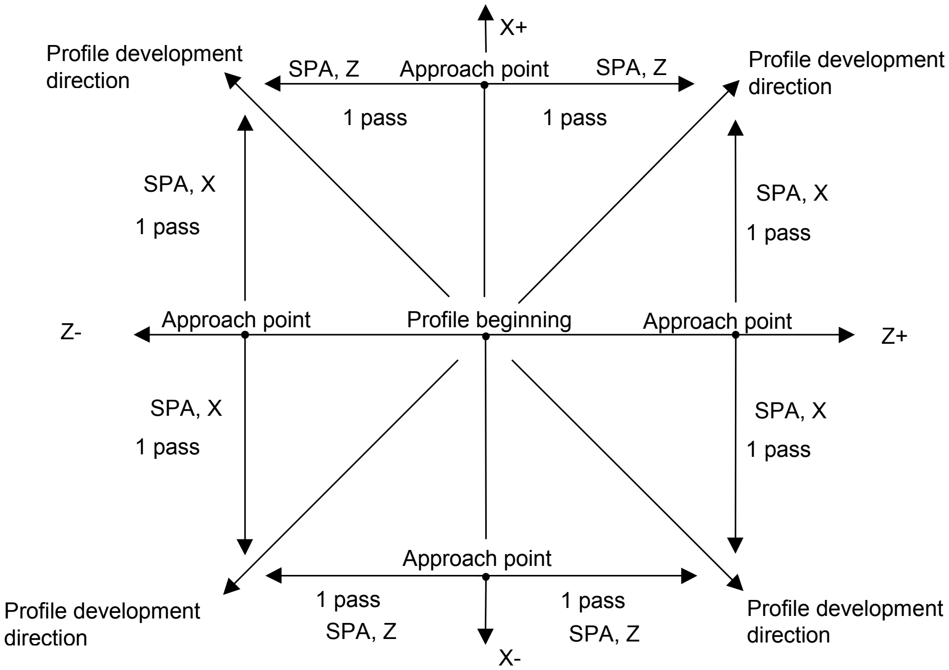


Figure 16-3 Example of roughing parallel to the Z axis

```
PROF2
p1 = Z0X30
I1 = p1, a180
I2 = Z-15X30, a120
I3 = Z-60X100, a150
I4 = Z-75X100, a180
p2 = Z-75X140
I5 = p2, a90
G21 p1
I1
I2
r15
I3
r18
I4
I5
G20 p2
```

```
..........
..........

T1.1 M6 S..F..
GX143 Z1.5
(SPA, PROF2, Z, 9, X1, Z1)
..........
..........
```

Diagram to determinate the profile development direction according to the approach point.



The profile development direction must be defined by taking into consideration the direction taken by the tool to perform the roughing cycle, which is also the profile development direction.

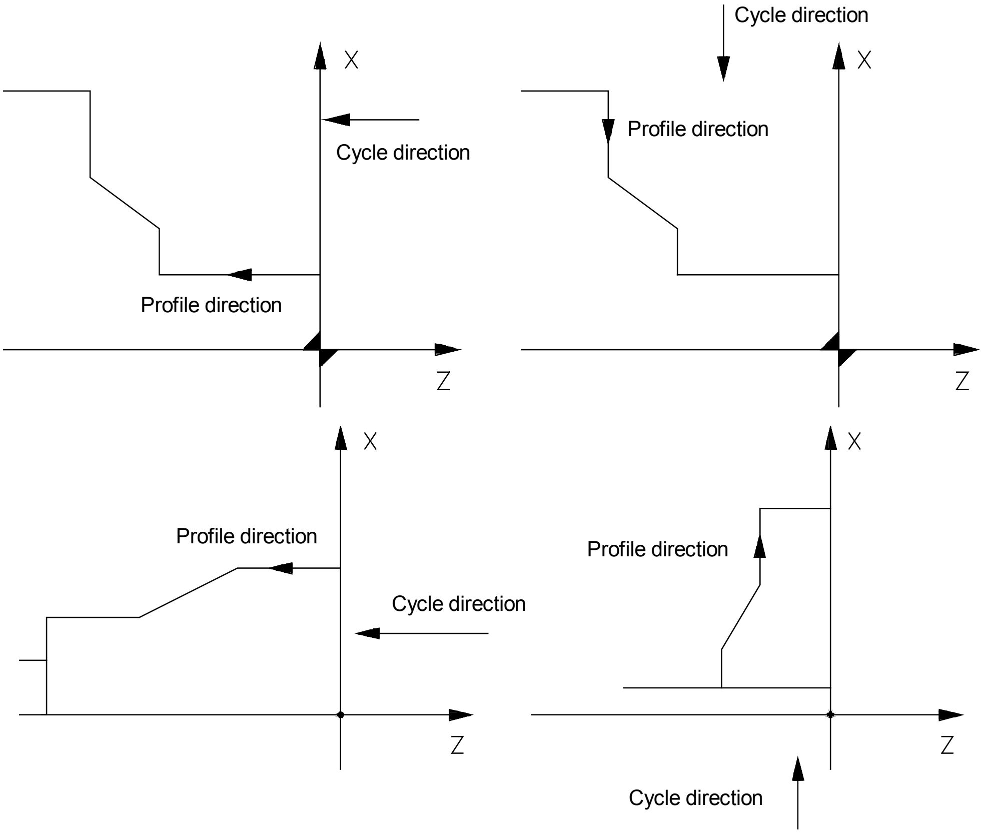


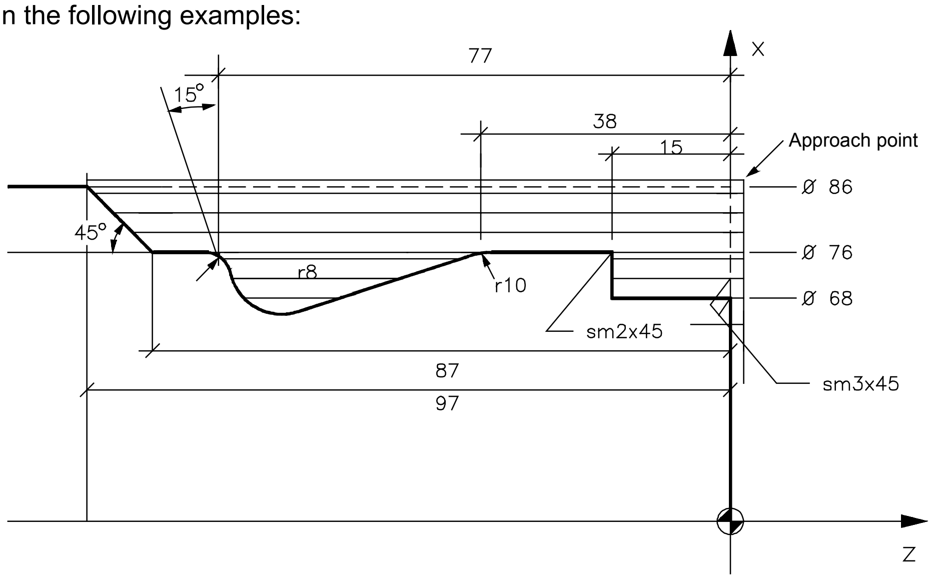
Figure 16-4 Tool profile and direction

MACRO-INSTRUCTIONS OF PARA-AXIAL ROUGHING WITH PRE-FINISHING

Roughing with cutting passes parallel to the ordinate and abscissa axis and final pass parallel to the profile:

(SPF, profile_name, axis_name, n_pass [,axis1 stock] [,axis2 stock])

For the meaning of the parameters and the chosen attach point , refer to the SPA macro-instructions. If the profile presents cavities, they will be rough-shaped during the SPF cycle as indicated in the following examples:



PROF3

```
p1 = Z0X62
l1 = p1, a135
l2 = Z0X68, a180
l3 = Z-15X0, a90
l4 = Z0X76, a180
l5 = Z-38X76, a198
l6 = Z-77X76, a105
p2 = Z-97, X86
l7 = p2, a135
G21 p1
l1
l2
l3
r2
l4
r10
l5
r-8
l6
r4
l4
l7
G20 p2
```

```
..........
..........

T1.1 M6 S..F..
GX90 Z2
(SPF, PROF3, Z, 7, X1, Z1)
..........
..........
```

IMPORTANT

The profile must be monotonous for the roughing axis or the system will indicates an error.
Therefore, the profile may present cavities but as regards the points trend along the roughing axis, it must be always increasing or always decreasing.

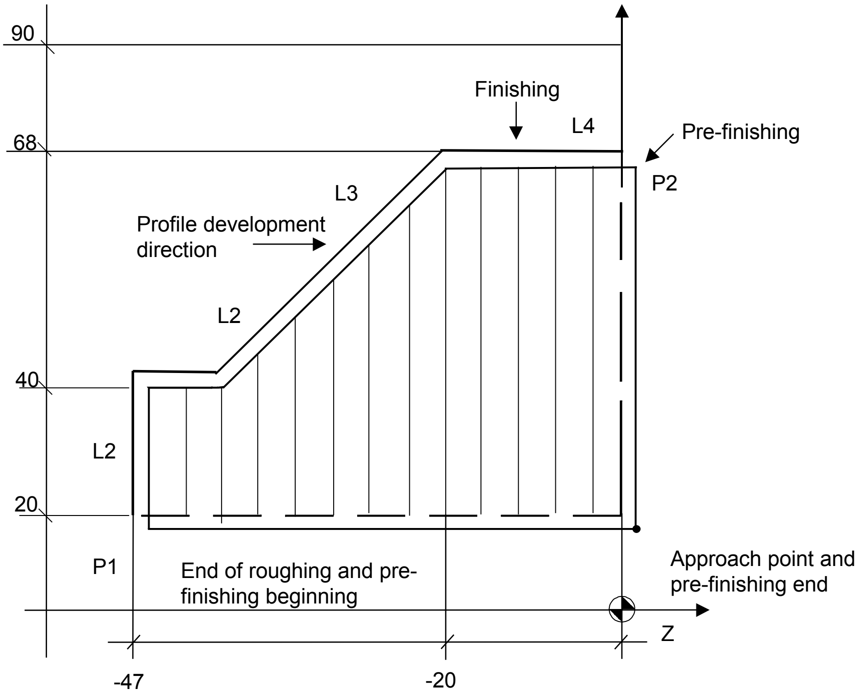

90
68
40
20
Finishing
L4
Pre-finishing
P2
L3
Profile development direction
L2
L2
P1
End of roughing and pre-finishing beginning
Approach point and pre-finishing end
Z
-47
-20

Figure 16-5 Example of internal para-axial roughing with pre-finishing parallel to the X axis

```
PROF4
p1 = Z-47X20
l1 = p1, a90
l2 = Z0X40, a0
l3 = Z-20X68, a45
p2 = Z0X68
l4 = p2, a0
G21 p1
l1
l2
l3
l4
G20 p2
```

```
..........
..........

T..S..F..
GX15 Z2.5
(SPF, PROF4, X, 10, X2, Z2)
..........
..........
(CLP, PROF4)
```

MACRO-INSTRUCTION OF ROUGHING PARALLEL TO THE PROFILE

This macro-instruction has been designed for roughing of pre-formed parts where the stock allowance is more or less constant on the rough stock.

The format is:

(SPP,profile_name, n_pass, axis_name, stock1, stock 2, axis_name, stock1, stock 2)

where:

profile_name	is an ASCII string representing the part program name in which the profile is described.
n_pass	number of the roughing passes. It may be a value between 2 and 255.
axis_name	it is the axis name (abscissa Z / ordinate X) on which the indicated stock must to be left.
stock1	stock to be left on the finished part in Z / X
stock2	stock on the rough part Z / X

The stock values must always be programmed, even when their value is zero are valid. The same considerations made for the SPA and SPF macro-instructions must be applied to the approach point.

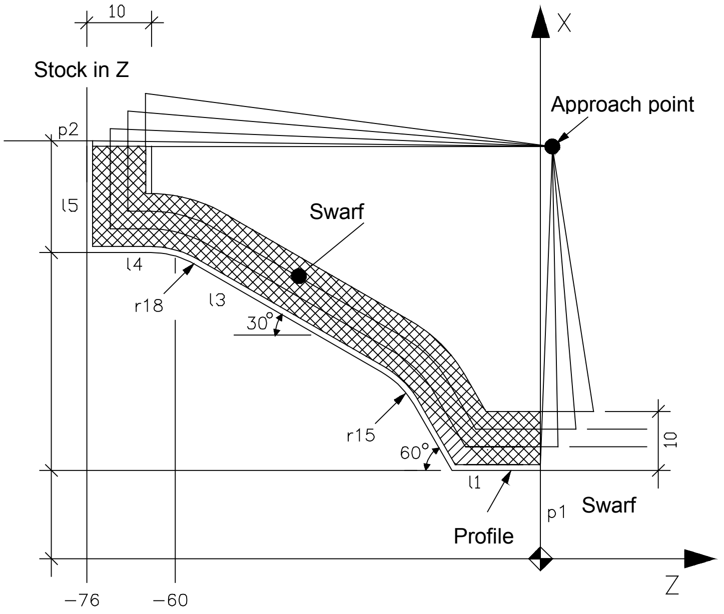


Figure 16-6 SPP use

```
..........
..........
T1.1 M6 S...F...M...
T.. S.. F.. M..
G X143 Z2
(SPP,PROF2,4,Z,1,10,X,1,10)
..........
```

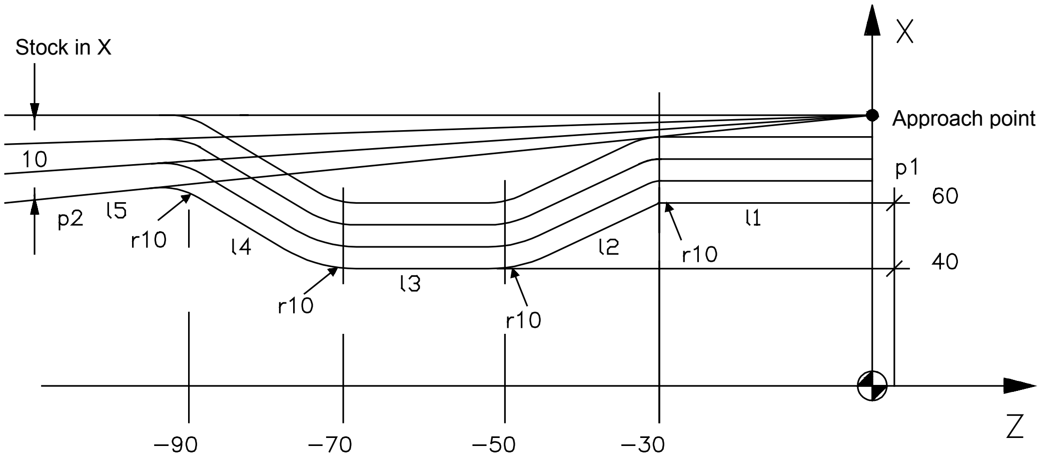


Figure 16-7

```
..........
..........
T.. S.. F..M..
G X84 Z1
(SPP,PROF5,4,X,0,10,Z,0,0)
..........
..........
```

PROF5

```
p1 = Z0X60
l1 = p1, a180
l2 = Z-30X60,Z-50X40
l3 = Z0X40, a180
l4 = Z-70X40, z-90X60
p2 = Z-110X60
l5 = p2, a180
G21 p1
l1
r10
l2
r-10
l3
r-10
l4
r10
l5
G20 p2
```

MACRO-INSTRUCTION OF A PROFILE FINISHING

The programming format is:

(CLP,profile_name)

where:

profile_name	is an ASCII string representing the part program name in which the profile is described.

CLP is the only profiling cycle during which any F functions programmed inside the profile are used.

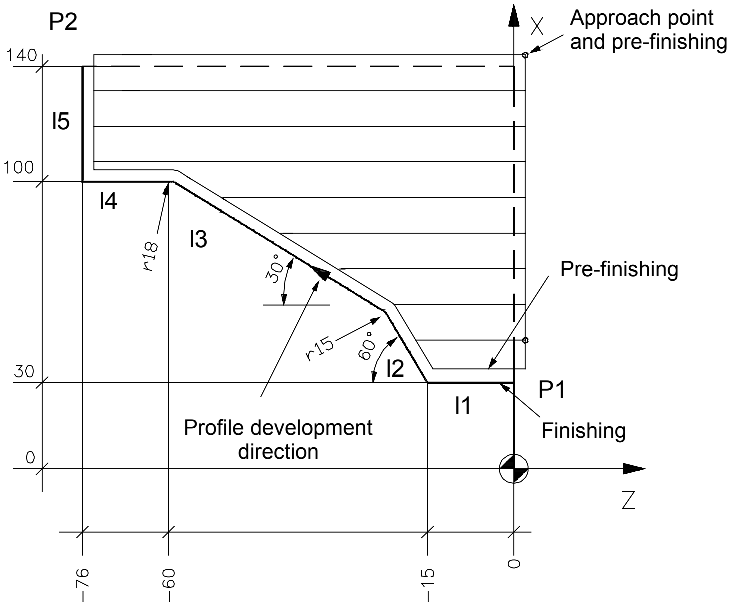


Figure 16-8 Para-axial roughing with pre-finishing and finishing

```
..........
T1.1M6 S..F..
GX143 Z1.5
(SPF, PROF2,Z,9,X1,Z1)
X300 Z200
T2.2 M6 S..F..
GX30 Z2
(CLP, PROF2)
GX300 Z200
..........
```

THREADING CYCLE

The threading cycle allows a cylindrical or taper thread programmed in a single block to be executed in several passes. The programming format is:

(FIL,axis1 , [axis2 ,] pass, n_ pass [,F .] [,R .] [,G ..] [,P...] [,M .] [,T ..] [,H ..] [,a ...] [,b..] [,r ..])

where:

axis1	abscissa and/or ordinate axis (normally Z or X)
axis2	name of abscissa and/or ordinate axis (normally Z or X) and final co-ordinate for the indicated axis (optional, it is programmed in tapered threading)
pitch	threading pitch
n_pass	number of threading passes. It may be a value between 1 and 255.
F...	number of polishing passes. It may be a value between 1 and 255. Default: 0
R...	distance of tool from return part (default value 1)
G...	internal/external threading: 0 = external thread 1 = internal thread. Default: external (0)
P...	starts number (default value 1)
M...	entry mode 0 = rough-shaping at 1 angle (side) 1 = rough-shaping at 2 angles (zig zag). Default: 0
T...	output mode: 0 = thread with final groove 1 = thread without final groove (tearing) 2 = output with circular interpolation (connection). Default: 0
H...	metric/inches/non-standard 0 = metric thread 1 = whitworth thread 2 = non standard thread with depth and angle defined by parameters a and b. Default: 0
a...	angle of non-standard thread
b...	depth of non-standard thread
r...	output radius for the exit thread. Obligatory if output mode 2 (T2) has been programmed.

Parameters included between square parenthesis may be omitted.

When non-standard threads are programmed (H2), the (K) pitch must be:

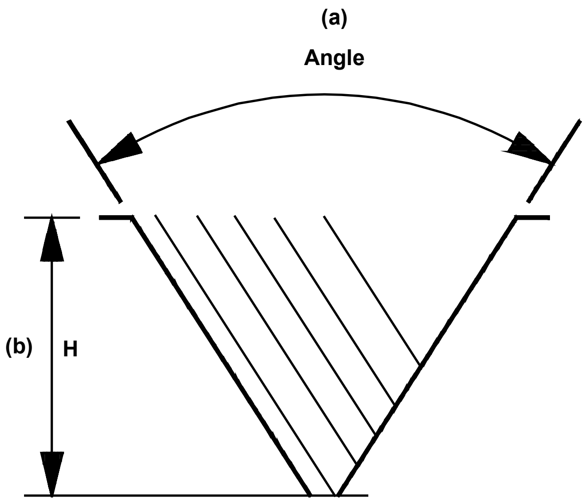
$$K > 2 * b * tg (a/2)$$

NOTES

a) The unit calculates the movements along the thread so that passes are performed with constatn swarf removal.

In case of threads with mulitples starts, the pitch to be defined is that of a single turn. The system performs each pass on all starts before executing subsequent passes. The management of multiple starts principles is carried out without moving the threads start point, but by introducing an angular displacement with respect to the zero angular position of the spindle.

b) For thread with a final groove, the theoretical final Z position must be programmed because the cycle will automatically stop and retract the axes , half a pitch after the theorical final position.The parameters **a** (thread angle) and **b** (thread depth) are required only for programming non-standard thread (H2).



	metric	whitw.	others
angle	60°	50°	a
H	f(pitch)	f(pitch)	b

Figure 16-9

Single start thread

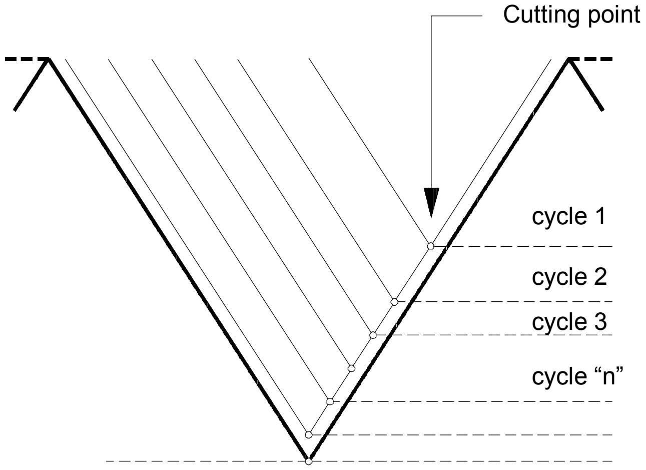


Figure 16-10

Two start thread

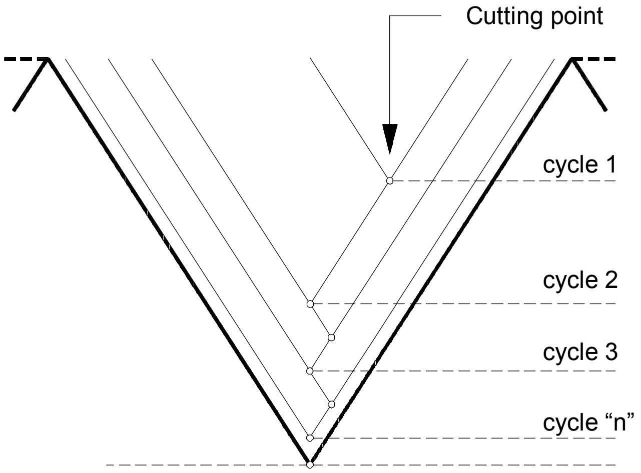


Figure 16-11

c) For threads without a final grove, the tool reaches the programmed endpoint and then reaches the return diameter with a tapered thread.

d) In threads without a final groove, SEMIAUTO must not be used, otherwise the result will be a final groove .

e) The threading cycle must not be defined in G28.

f) For taper threading, the maximum taper allowed is equal to 1/2 of the thread angle.

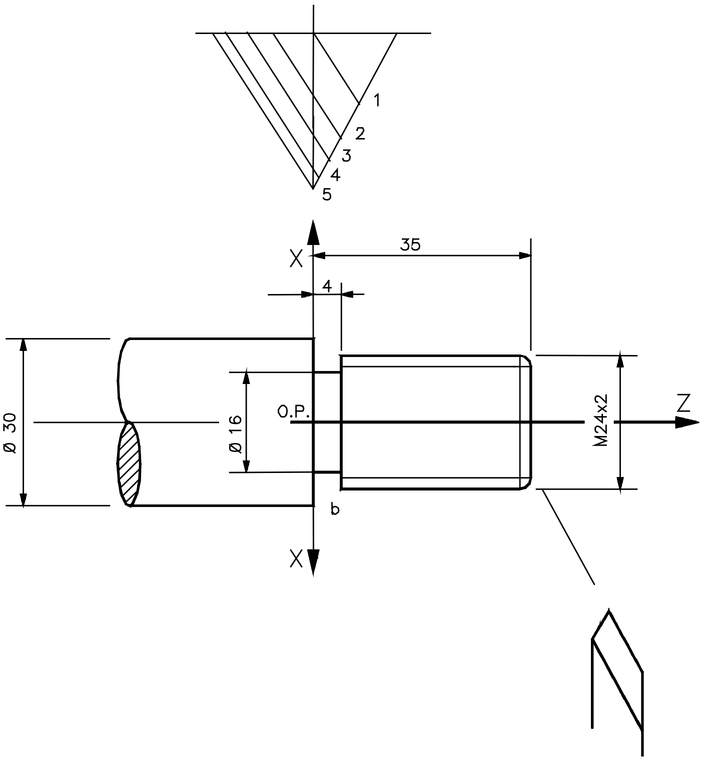


Figure 16-12 Example of threading cycle

```
N35  T5.5 M6
N36  G0 G97 X24 Z37 S250 M3 M8
N37  (FIL,Z4,2,5,F1,R2)
N38  G0 X250 Z215
```

GROOVE CUTTING CYCLE

The cycle generates a sequence to generate a groove parallel to the abscissa or ordinate axis (generally Z or X), internal or external.

The programming format to obtain a parallel groove to a plane axis is:

(TGL,axis1, axis2, width_tool [, coord_external,B/R..., B/R...,])

The minimum requested format is:

(TGL axis1, axis_2, width_ tool)

where:

axis1	Plane abscissa or ordinate axis name and final co-ordinate of the groove (the initial co-ordinate must be programmed before defining the groove cutting cycle)
axis2	Name of 2^{nd} axis of plane and internal position of the groove (see the drawing)
width_tool	Tool width
coord_external	Co-ordinate of top of grove.
B/R	Initial radius/bevel (optional)
B/R	Final radius/bevel (optional)

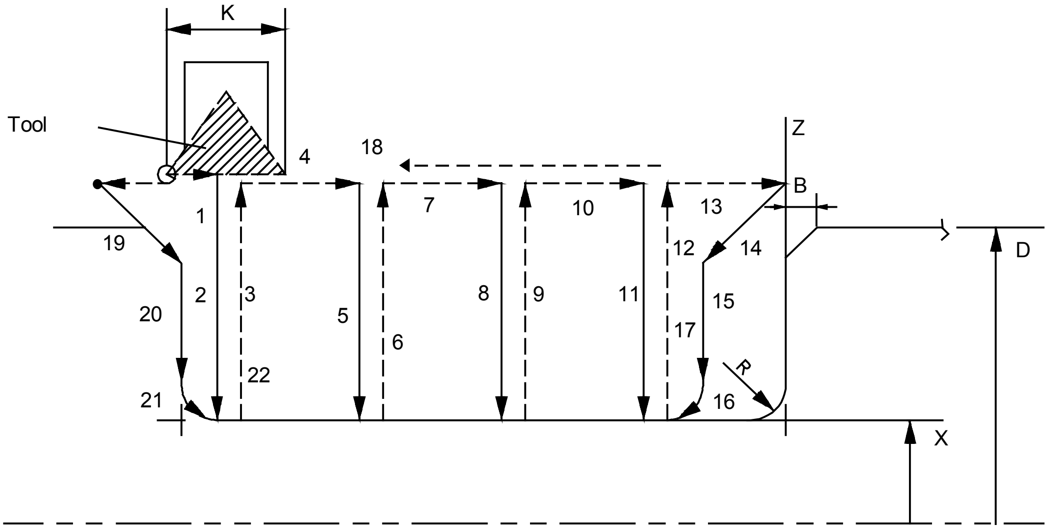


Figure 16-13 Groove cutting cycle

The TGL command must be preceded by a block of movement in G0/G1 on the cycle beginning point.

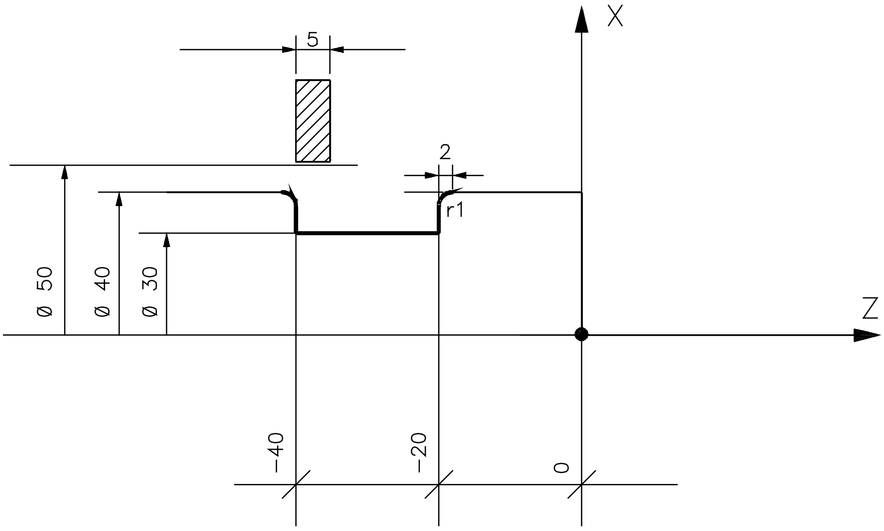


Figure 16-14 Example of groove parallel to the Z axis

Axis1	Z-20
Axis2	X30
Width-tool	5
Coord-external	40
B/R	R1,R0

```
N1 T1.1 M6 S.. F..
N2 G X50 Z-40
N3 TMR=2
N4 (TGL,Z-20,X30,5,40,R1,R0)
N5G X10 Z100
```

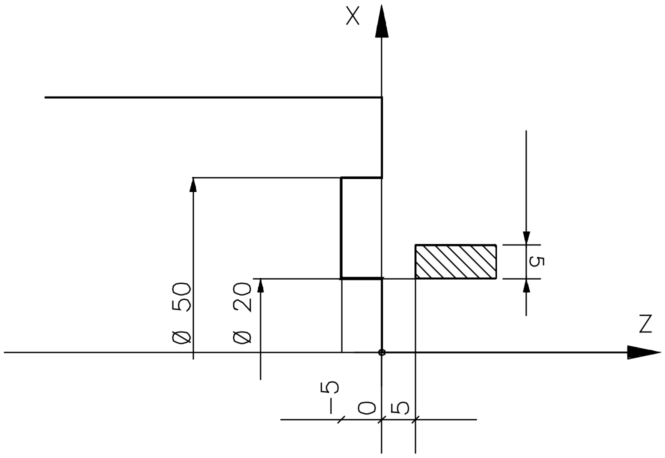


Figure 16-15 Example of groove parallel to the X axis

Axis1	X50
Axis2	Z-5
Width-tool	5

```
N1 T1.1 M6
N2 G X20 Z5
N3 TMR=2
N4 (TGL,X50,Z-5,5)
```

The systems automatically forces a dwell at the end of the groove.

The stop time is defined by the three-letter TMR. If a stop is not wanted, program TMR=0 before the groove cutting cycle.

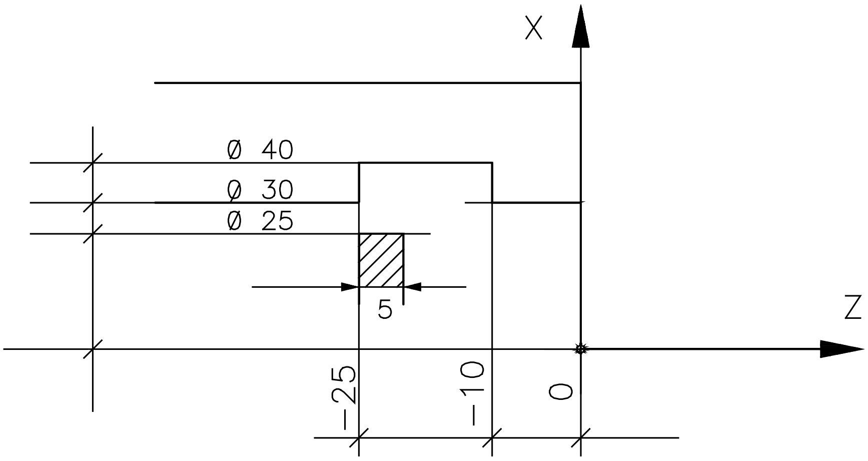


Figure 16-16 Internal groove

Axis1	Z-10
Axis2	X40
Width-tool	5

```
N1 T1.1 M6 S.. F.. M..
N2 G X25
N3 Z-25
N4 TMR=2
N5 (TGL,Z-10,X40,5)
N6 ..........
```

At the cycle end the tool returns to the cycle start point programmed in the previous block.

END OF CHAPTER

Chapter 17

FILTERS

GENERAL

Nowadays filters are used in the widest variety of applications, e.g., to regularise sets of values, to process data after eliminating a specific portion thereof, to analyse signals by cutting the high frequencies, or in the finite band pre-compensation of control loops.

This chapter describes the various types of filters that can be configured and applied in OSAI control units, designed to improve machine tool performance from both the geometric and dynamic point of view and hence the finished quality of the parts produced.

FLT – Filter programming

The FLT triliteral makes it possible to configure and enable filters that act on the geometry and the dynamic movement of the axes.

The FLT instruction, depending on the mode defined in the command, is used as a preparatory command (allowing to prepare the filter parameters) and as an executive command (activating and deactivating the filter).

Syntax

(FLT, *mode1, filter_id, name_axis [,parameters]***)**

or

(FLT, *mode2 [,filter_id [,axes_names]]***)**

where:

mode1 is the preparatory mode for the filter:

- **R** = parameter reading; it is possible to read the values of the parameters of a filter previously written with the same triliteral or configured in AMP. The parameters must be numerical variables, otherwise a format error is notified
- **W** = parameter writing; makes it possible to configure the filter, by writing the relevant parameters, but does not enable it: in this mode, the parameters can be both variables and numerals.

  N.B.: When a filter is configured for the first time, a “filter resource” is allocated, A total of 32 filter resources can be allocated for the entire CNC.

For this mode, the number of parameters programmed must be the same as the number of parameters managed by the filter specified in the triliteral

- **S** = filter status request: -1 = filter not configured

  0 = filter inactive

  1 = filter active

  The status is stored in the variable specified by the single parameter admitted by this command

mode2 Is the executive mode that can be performed on the filter :

- **E** = enable; permits the simultaneous activation of one or more of the filters on one or more axes of the process listed in *axes_names*

- **D** = disable; makes it possible to deactivate simultaneously one or all the filters on one or more axes of the process listed in *axes_names*
- **C** = delete: makes it possible to release a “filter resource” which is no longer required. This command can be used in the event of a resources exhausted error (-5) during the configuration of a new filter.

filter_id	Identification of the filter used. This is required if the filter is programmed in *mode1*. For further information on the value to be inserted in this field, see the paragraph “Types of filters that can be configured”
name_axis	Programmed with *mode1*: it is the name of a single process axis to which the filtering algorithm is to be applied. If the axis does not belong to the current process, the system issues a format error
axes_names	Programmed with *mode 2*: it is a list of the names of the process axes to which the filtering algorithm is to be applied. If the axes do not belong to the current process, the system issues a format error
parameters	To be used only in *mode1* programming, it is a list of the parameters associated with the filter. The number of parameters must be consistent with the number managed by the filter.

Characteristics:

Writing these parameters is permanent, i.e., the original value is only restored at next power-up, not with RESET.

An **FLT** command **on virtual axes** is not executed and therefore on this kind of axes the filter remains non configured.

Types of filter that can be configured:

Filter_id	***Filter type*** **Description**	**Parameters**
1	*Floating average on the command* It is possible, with this type of filter, to average all the points and speeds sent to the servo amplifiers during a configurable number of samples. This function makes it possible to improve the dynamic behaviour of the electromechanical system, in case of programming by points performed with low enough tolerances. By calculating a floating average of the points it is possible to round off the corners of the polygonal trajectory generated by CAD systems	Lower and upper sampling limits (**L**). The total number of points considered for calculation of the average reference value applied to the system will be 2*L+1: current point + L previous points and L subsequent points The **value of L** cannot exceed **32**

EXAMPLE 1:

Let us configure, on the first four axes of the process, a filter that calculates the floating average of the points, based on three samples.

```
(FLT, W, 1, X , E1)
(FLT, W, 1, Y , E1)
(FLT, W, 1, Z , E1)
(FLT, W, 1, A , E1)
```

E1 = L = 3

The average will be calculated on 7 reference values: current point + 3 previous and 3 subsequent points

EXAMPLE 2:

Let us assume you want to:

1) enable the filter of example 1 only on the linear axes => (FLT, E, 1, XYZ)

2) determine the enable status of the filter on the Y, A and B axes => (FLT, S, 1, Y, E2)
(FLT, S, 1, A, E3)
(FLT, S, 1, B, E4)

after the command execution, we shall get E2 = 1, E3 = 0, E4 = -1

3) enable the filter of example 1 on all the axes in the process => (FLT, E, 1)

4) disable filter 1 on all the axes in the process => (FLT, D, 1)

END OF CHAPTER

CHARACTERS AND COMMANDS

TABLE OF CHARACTERS

Following is the punched tape format of the ASCII, EIA and ISO characters recognized by the control. Note that, depending on the code, there may be odd parity (EIA), even parity (ISO) or no parity (ASCII) .

CHARACTER	DESCRIPTION
0 to 9	• Numbers
+	• Addition
-	• Subtraction
*	• Multiplication
/	• Division
.	• Decimal Point
"	• Label Identifier
(	• Open Parenthesis
)	• Close Parenthesis
[	• Open Parenthesis
]	• Close Parenthesis
{	• Open Parenthesis
}	• Close Parenthesis
;	• Comment Symbol
,	• Parameter Separator

CHARACTER	**DESCRIPTION**
=	• Assignment Symbol
<	• Less than (jump symbol)
>	• Great than (jump symbol)
LF (ISO or ASCII line feed) CR (EIA carriage return)	• Block End
#	• Synchronization
&	• Asynchronization
!	• Prefix User variable
@	• Prefix PLUS variable
>>	• Increase on single operand
A a	• Axis name • Acceleration on profile • Angle with GTL
B b	• Axis name • Bevel in cutter diameter compensation
C c	• Axis name • Circle with GTL
D d	• Axis name • Distance with GTL • Tool diameter (RQP,RQT)
E	• Parameters used in machining cycles
F	• Axes feedrate in G1-G2-G3
G	• Preparatory Code (G00 - G99) • Reserved (G100 - G299) • Paramacro subroutines (G300 - G999)
H h	• Parameters used in PARAMACROS • Offset change during continuous move
I i	• Coordinates of the arc centre in a circular interpolation (G2 G3) (abscissa) • Variable pitch in G33 • Tool dimension vector (with j and k)

CHARACTER	DESCRIPTION
J j	• Coordinates of the arc centre in a circular interpolation (G2-G3) (ordinate) • Min depth increase in (G83) • Tool dimension vector (with i and k)
K k	• Reduction factor for I and J in drilling cycle • Threading Pitch (G33) • Threading Pitch (G84) • Helix pitch in helical interpolation • Tool dimension vector (with i and j)
L l	• User table variables • Length 1 of tool offset (RQT, RQP) • Length 2 of tool offset (RQT, RQP) • Straight line with GTL
M m	• Auxiliary functions • Normal vector on surface (with n and o)
N n	• Part program block number • Normal vector on surface (with m and o)
o	• Normal vector on surface (with n and m) • Source with GTL
P p	• Axis name • Point with GTL
Q	• Axis name
R r	• Rapid positioning in cycles G81 - G89 • Deviation from the spindle zero (used in multi-start threads) adius in a circular interpolation G02-G03 • Profile radius (used for cutter diameter compensation only) G73 cycle • Radius with GTL
S s	• Spindle speed • Intersection with GTL
T t	• Tool and tool offset address • Time needed to complete the move in one block
U u	• Axis name • Compensation factors in axis 1 (offset)
V v	• Axis name • Compensation factors in axis 2 (offset)

CHARACTER	DESCRIPTION
W w	• Axis name • Compensation factors in axis 3 (offset)
X	• Axis name
Y	• Axis name
Z	• Axis name

G CODES

The table below lists the G codes available with 10 Series systems

G CODE	FUNCTION
G00	Rapid axes positioning
G01	Linear interpolation
G02	Circular interpolation CW
G03	Circular interpolation CCW
G04	Dwell at end of step
G09	Deceleration at end of step
G16	Defined interpolation plane
G17	Circular interpolation and cutter diameter compensation in the XY plane
G18	Circular interpolation and cutter diameter compensation in the ZX plane
G19	Circular interpolation and cutter diameter compensation in the YZ plane
G20	Closes GTL profile
G21	Opens GTL profile
G27	Continuous sequence operation with automatic speed reduction on corners
G28	Continuous sequence operation without speed reduction on corners
G29	Point-to-point mode
G33	Constant or variable pitch thread
G40	Disables cutter diameter compensation
G41	Cutter diameter compensation - tool left
G42	Cutter diameter compensation - tool right
G60	Closes HSM profile
G61	Opens HSM profile
G70	Programming in inches
G71	Programming in millimetres
G79	Programming referred to machine zero
G80	Disables fixed cycles
G81	Drilling cycle
G82	Spot-facing cycle
G83	Deep hole drilling cycle
G84	Tapping cycle
G85	Reaming cycle
G86	Boring cycle
G89	Boring cycle with dwell
G90	Absolute programming
G91	Incremental programming
G92	Axis presetting without mirror
G98	Axis presetting with mirror
G93	Inverse time (V/D) feedrate programming
G94	Feedrate programming in ipm or mmpm
G95	Feedrate programming in ipr per revolution or mmpr
G96	Constant surface speed in fpm or mpm
G97	Spindle speed programming in rpm
G72	Point probing with probe ball radius compensation
G73	Hole probing with probe ball radius compensation
G74	Probing for theoretical deviation from point without probe ball radius compensation
G99	Deletes G92

MATHEMATICAL FUNCTIONS

- **SIN**
- **COS**
- **TAN**
- **ARS**
- **ARC**
- **ART**
- **SQR**
- **ABS**
- **INT**
- **NEG**
- **MOD**
- **AND**
- **OR**
- **NOT**
- **FEL**
- **FEC**
- **FEP**

LOCAL AND SYSTEM VARIABLES

VARIABLE NAME	FUNCTION
L	Plus User Table variable
SN	System Number
SC	System Character
TIM	System Time
E	Local variable
H	Paramacro variable
HF	Paramacro flag variable
HC	Paramacro string variable
!name	User variable
@name	PLUS variable
VEF	Velocity Factor
TPO	Tool path optimization
TPT	TPO threshold
CET (PRC)	Circular Endpoint Tolerance
FCT	Full Circle Threshold
ARM	Defining Arc Normalization Mode
DLA	Enables/disables look ahead
MDA	Maximum Deceleration Angle Computation
DWT (TMR)	Dwell time
SSL	Spindle Speed Limit
ERR	Ables/Disables Part Program Errors handling
STE	System Error
MSA (UOV)	Defining a Machining Stock Allowance
TRP (RMS	Tapping Return Percentage
DSB	Disable slashed blocks
UPA (RTA)	Update Probe Abscissa
UPO (RTO)	Update Probe Ordinate
VFF	Velocity Feed Forward
ODH	Online Debug Help
DYM	Execution mode with G27

THREE-LETTER CODES

CODE	**FUNCTION**
DAN	Define axis name
IPB (DTL)	In Position Band
UAO	Use absolute origin
UTO (UOT)	Use temporary origin
UIO	Use incremental origin
RQO	Requalify origin
SOL (DLO)	Software overtravels
DPA (DSA)	Define protected areas
PAE (ASC)	Protected area enable
PAD (DSC)	Protected area disable
MIR	Mirror machining
ROT (URT)	Active plane rotation
SCF	Scale factor
AXO	Axis Offset Definition
RQT (RQU)	Requalifying Tool Offset
RQP	Requalifying Tool Offset
TOU (TOF)	Declare a tool out of life
DPP (DPT)	Defining Probing Parameters
LOA	Dual port loading of tables
RPT	Open repetition of a set of program blocks
ERP	Close repetition a set of program blocks
CLS	Call a subroutine for execution
EPP	Execute a portion of a part program (subprogram)
PTH	Set *path* for subroutines and paramacros
EPP	Execution of a part of a program
EPB	Execution of a program block
GTO	Branch Command
IF, ELSE, ENDIF	Conditional Execution of parts of a program

CODE	FUNCTION
GDV	Device Definition
RDV	Release Device
UDA	Dual Axes
SDA	Special Dual Axes
UGS (UCG)	Use the graphic scale
CGS (CLG)	Clear the graphic field
DGS (DCG)	Disable the graphic scale
UPR	Uses Plane Rotated
UVP	Use Virtual Polar
UVC	Use Virtual Cylindrical
TCP	Tool Center Point
DIS	Display a variable
DLY	Cause a delay in program execution
DSB	Disable slashed blocks
REL	Disactivate a part-program
WOS	Put the system on hold for a signal
GTA	Axes acquisition
GTS	Spindle sharing
SND	Send a synchronization message
WAI	Wait for a synchronization message
EXE	Automatic activation of a part program
ECM	Execution in MDI mode of a block in a specified process
PRO	Definition of default process
DCC	Definition of communication channel
PVS	PLUS variable selection
GTP	Determine approach point for automatic contour milling
CCP	Perform automatic contour milling

CODE	FUNCTION
SPA	Para-axial rough shaping without pre-finishing
SPF	Para-axial rough shaping with pre-finishing
SPP	Parallel rough shaping to the profile
CLP	Finishing cycle
FIL	Threading cycle
TGL	Groove cutting cycle

ASCII CODES

The tables that follow show the 256 elements of the extended ASCII character set, together with their decimal and hexadecimal equivalents.

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
000	00	BLANK (NULL) (NULL)	016	10	► (DLE)	032	20	BLANK (SPACED)	048	30	0
001	01	☺ (SOH)	017	11	◄ (DC1)	033	21	!	049	31	1
002	02	☻ (STX)	018	12	↕ (DC2)	034	22	"	050	32	2
003	03	♥ (ETX)	019	13	‼ (DC3)	035	23	#	051	33	3
004	04	♦ (EOT)	020	14	¶ (DC4)	036	24	$	052	34	4
005	05	♣ (ENQ)	021	15	§ (NAC)	037	25	%	053	35	5
006	06	♠ (ACH)	022	16	▬ (SYN)	038	26	&	054	36	6
007	07	● (BEL)	023	17	↨ (ETB)	039	27	'	055	37	7
008	08	◘ (BS)	024	18	↑ (CAN)	040	28	(	056	38	8
009	09	○ (HT)	025	19	↓ (EM)	041	29	)	057	39	9
010	0A	◙ (LF)	026	1A	→ (SUB)	042	2A	*	058	3A	:
011	0B	♂ (VT)	027	1B	← (ESC)	043	2B	+	059	3B	;
012	0C	♀ (FF)	028	1C	∟ (FS)	044	2C	,	060	3C	<
013	0D	♪ (CR)	029	1D	↔ (GS)	045	2D	-	061	3D	=
014	0E	♫ (SO)	030	1E	▲ (RS)	046	2E	.	062	3E	>
015	0F	☼ (SI)	031	1F	▼ (US)	047	2F	/	063	3F	?

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
064	40	@	080	50	P	096	60	`	112	70	p
065	41	A	081	51	Q	097	61	a	113	71	q
066	42	B	082	52	R	098	62	b	114	72	r
067	43	C	083	53	S	099	63	c	115	73	s
068	44	D	084	54	T	100	64	d	116	74	t
069	45	E	085	55	U	101	65	e	117	75	u
070	46	F	086	56	V	102	66	f	118	76	v
071	47	G	087	57	W	103	67	g	119	77	w
072	48	H	088	58	X	104	68	h	120	78	x
073	49	I	089	59	Y	105	69	i	121	79	y
074	4A	J	090	5A	Z	106	6A	j	122	7A	z
075	4B	K	091	5B	[	107	6B	k	123	7B	{
076	4C	L	092	5C	\	108	6C	l	124	7C	|
077	4D	M	093	5D	]	109	6D	m	125	7D	}
078	4E	N	094	5E	^	110	6E	n	126	7E	~
079	4F	O	095	5F	_	111	6F	o	127	7F	△

DEC	HEX	CHARACTER
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å

DEC	HEX	CHARACTER
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	Pt
159	9F	ƒ

DEC	HEX	CHARACTER
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»

DEC	HEX	CHARACTER
176	B0	░
177	B1	▒
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

DEC	HEX	CHARACTER
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧

DEC	HEX	CHARACTER
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	█
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀

DEC	HEX	CHARACTER
224	E0	α
225	E1	β
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	φ
233	E9	θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	∅
238	EE	∈
239	EF	∩

DEC	HEX	CHARACTER
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	•
250	FA	•
251	FB	√
252	FC	n
253	FD	²
254	FE	▮
255	FF	BLANK "FF"

Extended ASCII Character Set

END OF APPENDIX

Appendix B

ERROR MESSAGES

Description of error messages and remedial actions

Code	Message description and remedial action
NC001	**Syntax Error** Syntax error found in the part program block or in the MDI block
NC002	**Wrong number of axes for G code** This message is displayed to indicate that: • At least one axis must be programmed in G04 • Only one axis must be programmed in a canned cycle block (from G81 to G89).
NC003	**Canned cycle parameters missing** Canned cycle parameters (i.e. K, I, ...) are missing
NC004	**Missing parameters for G code** Parameters for G code are missing (i.e. G33 ...K)
NC005	**Missing J and/or K for G83 cycle** K or J parameter are missing in the G83 canned cycle
NC006	**Missing I and/or J for G2/G3 code** I and/or J parameters are missing in G2/G3 codes (circles)
NC007	**Probing cycle parameters missing** Probing cycle parameters (i.e., E or r) are missing
NC008	**Format error** This error is displayed in the following cases: • Wrong variable index • Feedrate (F) = 0 or negative • Wrong variable format • Repeat number is illegal (number of repetitions must be from 1 to 65535) • Format error in assignment, e.g. assignment to strings with different lengths • PLUS variables writing/reading error • Character variable format error in DIS code: not specified as CHAR • Protected area not allowed: 0< protected area number <4 • Variable not configured • Number of runs = 1 for triliteral SPP

Code	Message description and remedial action
NC009	**Undefined symbol** This message is displayed in the following cases: • Axis name not configured in AMP • Variable does not exist
NC010	**Overflow** Expression too long
NC011	**Function not allowed** This message is displayed in the following cases: • Activation of an M type "block calculation" requested with calculation block already active or on HOLD • Feedrate override with non linear active block requested
NC012	**Wrong use of axis slave** A slave axis previously declared in the UDA block is programmed directly in a part program or input in an MDI block. This error is also displayed in the following cases: 1. When an attempt is made to move the slave axis manually. 2. When programming UDA, if the slave axis is already involved in TCP programming as a linear or rotating axis. 3. When programming UDA, if the slave axis is already involved in virtual programming (UPR, UVP, UVC) as a real or virtual axis. 4. When programming UDA, if the master axis is already involved in virtual programming (UPR, UVP, UVC) as a virtual axis.
NC013	**Operand not allowed in canned cycle** Operand not allowed in G72 G73 G74 canned cycles
NC014	**K parameter not allowed in G84** K parameter not allowed during G84 programming of a spindle without transducer
NC015	**Wrong programming of G2/G3 code** Both the centre and the radius (R) of the circle are specified Eliminate either the radius or the IJ centre coordinates
NC016	**Illegal number of operands** Illegal number of operands in the AXO block
NC017	**Illegal number of pseudo axis** Too many pseudo axes programmed in the block (max. 6)
NC018	**Illegal number of axes in G33 code** More than 2 axes programmed in G33
NC020	**G not allowed** G not allowed in the threading cycle
NC021	**Operand not allowed with G code** Operand incompatible with type of movement

Code	Message description and remedial action
NC022	**Block and system state not congruent** • UDA enable/disable while other virtual mode is active • Synchronisation while blocks in execution are suspended. For example, synchronisation with cutter diameter compensation active. • Attempt to execute an MDI block while a macro is active
NC024	**G and program state not congruent** This message occurs when • G41-G42 canned cycles cannot be programmed when cutter diameter compensation is active • Threading cannot be programmed when cutter diameter compensation (G41 G42) is active or when a canned cycle is programmed • Interpolation planes (G17, G18, G19) cannot be programmed when cutter diameter compensation (G41 G42) is active
NC025	**G and dynamic mode not congruent** G function not compatible with current dynamic mode. For example: functions G72, G73, G74 do not accept in continuous mode (G27, G28) Switch from G27 to G28 or vice versa with active non-linear ramps (MOV > 1).
NC026	**G41/G42 and part program state not congruent** Cutter diameter compensation (G41/G42) not compatible with current program state.
NC027	**G needs spindle with transducer** G33 and the threading macro-cycle FIL need a spindle with transducer.
NC028	**G not congruent with feedrate mode** G72, G73, G74 must be executed when G94 is active
NC029	**Operand and part program state not congruent** Operand incompatible with current part program state For example: r, b operands are not allowed in the ISO standard state (G40)
NC030	**M and dynamic mode not congruent** Machine logic operands incompatible with active dynamic mode For example: M at motion end not compatible with (G27-G28) T programmed with G41/G42 active
NC031	**M/T/S and motion type not congruent** Machine logic operands incompatible with the type of move For example: G33 + end of motion M function
NC032	**Probing cycle operands inhibited** Probing cycle operands not allowed For example: Operands I,J,K,R,u,v,w,b,t are not allowed in G73 Operands I,J,K,R,u,v,w,b,t,r are not allowed in G72

NC033	**Missing third axis for helix** The third axis for helix programming is missing
NC034	**"Expedite" function without motion** An "expedite" M is present a block that does not program a move; "expedite" M's must always be associated motion to a move
NC035	**Feed or speed not programmed** • Feedrate or speed not programmed for canned cycle execution • Motion block in G1/G2/G3 without programmed feed
NC036	**Z-axis not found for G87 cycle** The z-axis has not been programmed for the cycle G87.
NC037	**Read only variable** The specified variable is of the read-only type. For example: TIM.
NC038	**Part program record too long** The programmed record has more than 127 characters. It is displayed in conjunction with the PART PROGRAM NAME message.
NC039	**Part program access denied** The part program file specified with this error message is not read-accessible as it is open in write mode for another user (e.g. Editor, DOS real time, etc.).
NC040	**P.P. block not allowed from serial line** Block not allowed during part program execution from serial line.
NC041	**Wrong serial line configuration for EPS**
NC042	**Nesting of IF greater than 32** The maximum number of nested IF commands has been exceeded.
NC043	**ELSE not allowed** An ELSE command has been programmed without a previous IF command.
NC044	**ENDIF not allowed** An ENDIF command has been programmed without a previous matching IF command
NC048	**Illegal argument for TAN** The argument of the TAN operator is 90 degrees (the result would be infinite)
NC049	**Illegal argument for SQR** The argument of the SQR operator (square root) is a negative number
NC050	**Too many programmed axes** More than 9 axes have been programmed in the block
NC051	**Division by zero** A division by zero has been detected in the expression that calculates an axis dimension (e.g. X10/0)

Code	Message description and remedial action
NC052	**String too long** The max. string length can be 80 characters. This message is displayed when a longer string is used in the following cases: • display of a string with the DIS code • string variable (SC) assignment
NC053	**Label duplicated** This message is displayed when the program is selected or activated. It shows that there are two identical labels in the part program. The duplicated label is also displayed.
NC054	**Undefined label** The label programmed in a branch instruction (GTO) or in a call for a subroutine (EPP) does not exist
NC055	**Label too long** This message is displayed when the system reads an SPG block. It indicates that a label having more than 6 characters has been programmed. The illegal label is also displayed
NC056	**Program table overflow** This message is displayed when the program is selected or activated. It indicates that the number of CLS for subroutines overflows the maximum configured in AMP. You can alter this parameter in AMP with the procedure described in the PROCESS CONFIGURATION section.
NC057	**Label table overflow** This message is displayed when the program is selected. It indicates that the number of programmed labels overflows the maximum configured in AMP. You can alter this parameter in AMP with the procedure described in the PROCESS CONFIGURATION section.
NC058	**End of program** End of file marker for: • block skipping • block editing • string search • program execution
NC059	**Beginning of program** Program start marker for: • block skipping • string search
NC060	**Nesting of RPT greater than 5** RPT max. nesting level (5) overflown
NC061	**Nesting of subroutine greater than 4** Subroutine max. nesting level (4) overflown
NC062	**Nesting of EPP greater than 5** EPP max. nesting level (5) overflown

Code	Message description and remedial action
NC063	**RPT/EPP cycle open at end of file** This message is displayed when: • The end of the file has been reached without finding the (ERP) block that closes the programmed (RPT) cycle • The end of the file has been reached without completing the subroutine defined with (EPP)
NC064	**ERP without RPT** (ERP) has been programmed without previously programming (RPT)
NC065	**Error during part program file handling** This message occurs during program reading/writing to indicate that: • a part program block has been skipped • error in program SPG/REL • error in subroutine opening/closing management • the accessed file does not exist or is protected
NC066	**Part program not found** The selected part program/subroutine is not stored in the E:\UPP directory
NC067	**Part program not selected** This error occurs when: • CYCLE START is given in AUTO but no part program has been selected • SKIP, MODIFY, ESCAPE commands are given but no part program has been selected • a branch instruction (GTO) is executed by the system in MDI mode
NC068	**Process number cut of range** The process number written in the three-letter code cannot be higher than the one set in AMP or less than 1.
NC069	**Paramacro modal already active** A paramacro is programmed when a modal paramacro is already active
NC070	**Paramacro not configured** The programmed paramacro has not been configured in AMP
NC078	**Software option not installed**
NC079	**Software option not available. Check security**
NC080	**Axis not referenced** This message occurs when: • The programmed axis is not referenced • The axis specified in the definition of a protected area with DPA is not referenced • The offset to be preset/requalified is associated to a non referenced axis
NC081	**Undefined DPP for probing cycle** Probing cycle parameters (approach coordinate, safety distance, velocity) are not defined in the DPP block

Code	Message description and remedial action
NC082	**Too many "Expedite" M codes** More than one expedite M code has been programmed in the block
NC083	**Undefined M code** The programmed M is not configured in AMP. Configure the M in AMP and restart the system
NC084	**Circle not congruent** The circle is not geometrically congruent: the radius or the final points are not correct
NC085	**Wrong threading parameters (I, K, R)** The programmed threading parameters (I, K and R) are not allowed. Calculate the I parameter with the following formula: $\frac{16\ k}{2(\text{threading distance})}$
NC086	**Helix pitch not congruent** The helix pitch is not geometrically correct
NC087	**Axes of plane needs same scale factor** Plane axes in G02/G03 programming (circle) must have the same scale factor. Change the scale factor with an SCF instruction
NC088	**Profile not congruent** The programmed ISO-offset profile is not correct
NC089	**Wrong direction on profile** Offset value in G41-G42 reverses the tool path direction
NC090	**Err. disabling cutter compensation** Wrong exit from cutter diameter compensation (G40)
NC091	**Too many blocks to resolve** Too many extra plane moves programmed with cutter diameter compensation active (G41-G42) (max. 2 extra plane moves).
NC092	**Entry in safety zone** The programmed move enters one of the three safety areas
NC093	**Canned cycle on rotate plane** Canned cycle programmed on rotated plane Disable plane rotation
NC094	**Canned cycle data not congruent** The parameters specified in the canned cycle (I, J, K, R) are not allowed. For example: canned cycle K = 0 A G84 or a G86 cycle is being performed with the spindle in non exclusive status.
NC095	**Missing parameters for G87** There are parameters missing in the G87 fixed drilling cycle . This cycle is used in the WOOD macros.

Code	Message description and remedial action
NC096	**Wrong probing cycle programming** This message appears when: • probing approach distance is null • hole probing is programmed with null radius (for example G73r0E5)
NC097	**Hole probing cycle not complete** The hole probing cycle not complete has not been completed
NC098	**Probing cycle not executed** This message occurs when the probe does not find the point to be probed before reaching the safety zone
NC099	**Probe has not been retracted** When measuring cycle starts the probe is already touching the part surface
NC100	**Hardware overtravel** The programmed axis has overflown the hardware overtravel. Jog it back within hardware travel limits
NC101	**Positive software overtravel** The programmed move causes the axis to exit the programmed or configured positive software travel limits
NC102	**Positive hardware overtravel limit** This message appears if the axis is jogged in the positive direction after it has reached its positive hardware overtravel limit. Select JOG DIR - and press CYCLE START to jog the axis back within the positive overtravel. **NOTE:** there is not other way of returning an axis to the HW operating limits
NC103	**Negative hardware overtravel limit** This message appears if the axis is on the programmed or configured negative hardware overtravel limit and you try to further jog it in the negative direction.
NC104	**Positive software overtravel limit** This message appears if the axis is on the programmed or configured positive hardware overtravel limit and you try to further jog it in the positive direction.
NC105	**Negative software overtravel limit** The axis is on the negative SW overtravel limit and we set a JOG DIR move
NC106	**JOG past software overtravel limit** The JOG INCR value would take the axis past the software overtravel limit
NC107	**Axes not on profile** This message appears if we try to quit CYCLE STOP after a series of jog moves without taking the axes back to the profile. Select JOG RETURN and return the axes to the profile
NC108	**Home and JOG DIR not congruent** This message appears when we try to home an axis in a JOG DIR opposite to the configured homing direction. **NOTE:** if the homing cycle is configured as automatic the system will automatically correct JOG DIR without displaying the error. Press the JOG DIR softkey to align the jogging direction to the configured homing axis direction

Code	Message description and remedial action
NC109	**Error in exit HOLD: mode changed** This message occurs when we try to exit from HOLD by setting an operating mode (BLK_BLK, AUTO, MANUAL) that is different from the one in which the system went on HOLD. Select the correct mode and re-try.
NC110	**Block not allowed in HOLD** This message occurs when: • we try to execute an MDI motion block with the system on HOLD. When the system is on HOLD axes can only be jogged. • the programmed M is configured as not allowed on HOLD
NC111	**Active reset denied** This message occurs when we tried to execute an ACTIVE RESET in the following conditions: • while a block is executed with G27-G28 • during execution of a block followed by a circular block (G02/G03) • during execution of the last block before a syntactically inappropriate block The system only accepts another ACTIVE RESET (particularly convenient for bypassing the circular block) or RESET
NC112	**Wrong use of roll-over axis with G90** The programmed coordinate for the axis with rollover in G90 is greater than the roll over pitch configured in AMP
NC113	**Wrong JOG DIR for jog return** If the jog direction is negative during automatic or manual JOG RETURN, the system forces positive jog direction. This message appears if reversal is prevented by the machine logic
NC115	**Probing cycle executed before the end of approach movement.** Probing cycle carried out during fast probe approach
NC116	**Wrong use of real axis during a virtualization modality** This message occurs when a real axis is programmed when virtual mode is active
NC117	**Tool direction active: movement not permitted** This message occurs when only the tool direction is active and all other movements are not allowed.
NC118	**Negative software overtravel** The programmed move causes the axis to move past the programmed or configured software negative travel limits
NC119	**Command not allowed during search in memory** Command not allowed during the search in memory

Code	Message description and remedial action
NC120	**Mode to select out of range** This message occurs when the selected mode is out of range. Allowed modes are in the 1-8 range: 1 MDI 2 AUTO 3 BLOCK by BLOCK 4 CONTINUOUS JOG 5 INCREMENTAL JOG 6 RETURN ON PROFILE 7 HOMING FILE 8 HPG
NC121	**Axes number to select out of range** The number of axes selected for manual moves with library call NC NC_SELAXI is out of range. The ***allowed range*** is from 1 to the number of axes configured for the process 1 < ***allowed range*** < n. of configurated axes +1
NC122	**Too many axes selected for manual move** A larger number of axis names than accepted have been inserted in the part program block. Edit the part program block.
NC123	**Bad select mode for cycle** This error is displayed when CYCLE START is pressed in the following conditions: • a mode other than MDI has been selected during execution of a tool change axis move • system on HOLD, AUTO or BLK/BLK with MBR (multiblock retrace) not configured in AMP • system on HOLD with MBR active and selected mode other than AUTO or BLK/BLK • system in IDLE and ACTIVE_RESET with selected mode other than AUTO or BLK/BLK • system in IDLE with MBR active and selected mode other than AUTO or BLK/BLK • system in HRUN with MBR active and selected mode other than AUTO or BLK/BLK • ACTIVE RESET command in HOLD status with selected mode other than MDI, AUTO or BLK/BLK. **NOTE:** For further information about the machine status (HOLD, MDI, HRUN, etc.) refer to the USER GUIDE.
NC124	**Wrong axis name** This error is displayed in the following when: • The name of the selected axis is not configured in the axes table associated to the process • The definition of the interpolation plane is not correct because its axis/axes are not configured in the axes table. • The plane to be defined with G17, G18, G19, G16 cannot be defined because one of the specified axes is not configured in the axes table • The axis specified in the NC_ACTUALOFS call does not exist • The axis specified in the SCF, MIR three-letter blocks is not configured • An axis coordinate reading error has occurred because the specified axis does not exist

Code	Message description and remedial action
	• The axis specified in the SOL, DPA, UDA, UGS, AXO, UAO three-letter blocks is not configured or is duplicated • the axis specified in the AX_SHARE Library call does not exist • you are releasing an axis shared with the logic through the GTA command.
NC125	**Data length out of range** The keyboard buffer for MDI blocks has been overflown. Allowed entry length ranges from 1 to 127 characters
NC126	**Failed to write variable** Value of variable not written
NC127	**Failed to read variable** Value of variable not read
NC128	**Operative limit definition wrong** • Error in defining the software operating limits with the three-letter mnemonic SOL. • The programmed software limits must be defined in configured software. • Software operating limits are not configured in AMP.
NC129	**Protected area not defined** This message occurs when you try to enable with a PAE a protected area which does not exist. Define a protected area with PAE.
NC130	**Offset length not defined for the axis** This message occurs when you try to preset or requalify an offset that is not associated to the specified axis.
NC131	**Tool orientation code wrong** The specified tool orientation code is illegal
NC132	**Error from PLUS environment** Error in the PLUS environment generated by PLUS library calls PL_SET92, PL_RESG92, PL_PRESCOR, PL_UAO, PL_UTO, PL_UIO, PL_RQT, PL_RQP, or PL_RQO Error during execution of: RQO, UAO, UTO, UIO, RQT, RQP, G92, GTS
NC133	**Error from servo environment** Error in the SERVO environment during origin or offset presetting The error can also be caused by the IPB command when the In Position Band value transferred to an axis is rounded out, in the internal computations, to less than 1 digit.
NC134	**Manual movement not executed , no axes configured** Manual movements are not allowed because no axes have been configured
NC135	**Axis not configured** The id programmed in the GTA or GTS three-letter code is not configured A non spindle axis has been programmed in the GTS three-letter code
NC136	**Programmed id identifies an auxiliary axis** The ID programmed in the GTA block corresponds to an auxiliary axis and is not allowed

Code	Message description and remedial action
NC137	**Axis or spindle not available** • The ID of the axis programmed in the GTA block corresponds to another process • ID of axis programmed in AX_SHARE Library function is not available. • Spindle axis requests with GTS cannot be accepted.
NC138	**Axis id duplicated** The axis ID is duplicated in the GTA block
NC139	**Programmed ID identifies a spindle** The ID programmed in the GTA block corresponds to a spindle axis and is not allowed
NC140	**Set spindle speed failed** The machine logic (task $SPROG) does not accept the variation of spindle speed.
NC141	**New tool request failed** The machine logic (task $nTPROG) does not accept the T code programming.
NC142	**M executed failed** The machine logic (task $mDECOD) does not accept the M code programming
NC143	**Pseudo axes programming failed** The machine logic (task $nPSEUDO) does not accept the pseudo axes programming.
NC144	**Axis motion inhibited** Axis motion denied by the machine logic (task $nCONMOV).
NC145	**End of move failed** The machine logic answers with error on the end of move signal (task $nENDMOV).
NC146	**Too many blocks without motion in continuous mode** Too many blocks without motion have been programmed in continuous mode (G27, G28)
NC149	**Program already selected** This error occurs in MDI mode when you try to activate the same paramacro several times.
NC150	**Axis homed** This message indicates that the axis has been homed.
NC151	**Axis on profile** This message indicates that RETURN TO PROFILE has successfully terminated and the axis has returned to the profile.
NC152	**End of automatic return to profile** This message indicates that automatic RETURN TO PROFILE has successfully terminated and all the axis have returned to the profile.
NC153	**End of block retrace** This message occurs when backward multiblock retrace. To retrace a greater number of blocks, alter the configured maximum.

Code	Message description and remedial action
NC156	**End of search in memory** End search in memory
NC160	**Command and system state not congruent** This message indicates that the command is not allowed in the present system status
NC161	**Internal error: class not exist**
NC162	**Internal error: NC message error** Switch the control off and then on again. If the message is retained, contact technical services.
NC190	**Insufficient length for tapping cycle** This error occurs when the distance covered in the acceleration and deceleration phases of the canned tapping cycle without transducer is longer than the total distance, and no space is left for machining.
NC191	**Insufficient length for tapping cycle with transducer** This error occurs when the distance covered in the acceleration and deceleration phases of the canned tapping cycle with transducer on the spindle is longer than the total distance, and no space is left for machining.
NC192	**Insufficient length for threading cycle** This error occurs when the distance covered in the acceleration and deceleration phases of the canned threading cycle is longer than the total distance, and no space is left for machining.
NC199	**Spindle not activated**
NC200	**File access error** Error in reading or writing a file.
NC201	**Set up file loading error** The axes configuration in the file accessed is different from the configuration on dual port.
NC202	**File/Dual port config. mismatch** The axes configuration in the file accessed is different from the configuration on dual port.
NC203	**Warning: table locked read only** PLUS denies access to the table on dual port.
NC204	**Illegal file size** The table on file is of wrong size.
NC205	**Empty magazine** The selected magazine doesn't have defined pockets.
NC206	**Pocket is still busy** The pocket defined for a tool is already reserved to a different tool.

Code	Message description and remedial action
NC207	**Illegal previous pocket** A tool taking up more than one pocket interferes with the pocket occupied by another tool (previous pocket).
NC208	**Illegal following pocket** A tool taking up more than one pocket interferes with the pocket occupied by
NC209	**Illegal random** An illegal random class has been traced in memory.
NC210	**Tool table is full** Dual port full during the loading of a tool table related to a certain magazine.
NC211	**Illegal double format for editor** A variable format non accessible to editor has been traced.
NC212	**Illegal magazine number into file** Error in reading or writing
NC213	**Pocket not initialized**
NC214	**Pocket not compatible**
NC215	**Illegal table name** The name of the table to be loaded is invalid. Make sure the extension of the table name is one the following: .TOL .USR .MAG .OFS .ORG .SPN
NC220	**Process undefined** The process has not been defined or configured. Define the default process with the PRO command or select an existing process for synchronisation commands.
NC221	**Wrong process type** A communication channel unsuitable for the command set has been used. Example: channel type 2 (PLUS) for EXE command execution.
NC222	**Wrong process number** The process number specified for synchronisation commands identifies the current process
NC223	**Process queue is full** The process queue (local or remote) that a message was sent to is full.
NC224	**Data sending too long** Data to be transmitted with SND are longer than 174 characters
NC225	**Data loading failed** The type or number of data transmitted with SND is not allowed

Code	Message description and remedial action
NC226	**Message already exists in queue** A SND command towards a process has been given before the process cleared the previous message.
NC227	**EXE or ECM failed** This message occurs when: • The status of the process to which the EXE or ECM command is sent does not allow automatic part program execution commands (RUN, HRUN, RUNH, HOLD) or an MDI instruction. • There is a syntax error in the program to which the EXE command is addressed
NC290	**Program activation denied** The machine logic has refused the activation of a part program.
NC291	**Program deactivation denied** The machine logic has refused the release of an active program.
NC292	**Axis acquisition request denied** The machine logic has refused the acquisition of some axes during the execution of the GTA three-letter code.
NC293	**Axis release request denied** The machine logic has refused the release of some axes during the execution of the GTA three-letter code.
NC294	**Spindle acquisition or sharing request denied** The machine logic has refused the acquisition or sharing of a spindle through the GTS three-letter code.
NC295	**Spindle release request denied** The machine logic has refused the release of a spindle through the GTS three-letter code.
NC296	**Spindle sharing change request denied** The machine logic has refused the spindle sharing status change through the GTS three-letter code.
NC297	**Fixed cycle not possible with shared spindle** A G84 or a G86 cycle has been programmed with the spindle set on non exclusive mode.
NC320	**UPR programming not allowed** UPR cannot be programmed when another virtual mode is active. This error is also displayed when: • One of the physical axes turns out to be SLAVE in UDA/SDA programming. • A type 5 or lower case UPR is programmed and no previous UPR is active. • The !R73 MODE user variable is set on 1 and the origins on the rotary axes are programmed in a UPR, or a type 5 UPR is programmed. • The origins on the rotary axes are programmed in a type 0, 1 or 10 UPR.
NC321	**Wrong incremental UPR programming** Incremental UPR can only be programmed if UPR is active.

Code	Message description and remedial action
NC322	**UPV programming not allowed** UPR cannot be programmed when another virtual mode is active. This error is also displayed if one of the real axes is a slave in UDA/SDA programming.
NC323	**Wrong axis type on UPV programming** The programmed type of real axis is not compatible with the virtual mode.
NC324	**Wrong programmed radius value** The radius programmed in the UVP block is not compatible with the linear axis position.
NC325	**UVC programming not allowed** UVC cannot be programmed when another virtual mode is active. This error is also displayed if the real axis is a slave in UDA/SDA programming.
NC326	**Programmed TCP code value out of range** The code that enables TCP is illegal.
NC327	**TCP programming not allowed** (TCP,5) cannot be programmed when another virtual mode is active. This error is also displayed if one of the linear or rotating axes of the TCP is a slave in UDA/SDA programming.
NC328	**TCP programming not congruent** The request to enable TCP is not compatible with the current TCP mode.
NC329	**Error on tangential TCP activation** Error during (TCP,4) enable. Check whether the specified axes ID's are configured in the user table.
NC330	**Error during get or release axes** GTA cannot be enabled when offsets, canned cycles or a virtual mode are active.
NC331	**Axis interpolator clock not congruent** One or more axes forming the object of a GTA or GTS command have been characterised with an interpolator clock different to that of the current process.
NC332	**Zero value of ijk module** ijk error programming with active TCP: the module with such values is equal to zero.
NC333	**Wrong programming of ijk, mno** ijk and/or mno wrong programmed.
NC334	**Number of contouring blocks overflow** The max. number of blocks defined in AMP for automatic contouring or for rough-machining cycles is lower than required.
NC340	**Circles/lines not defined** The circle/line programmed in the GTL profile has not been defined.

Code	Message description and remedial action
NC341	**Wrong definition of circles/lines** There is an error in the definition of a GTL circle/line.
NC342	**Circles/lines not intersecting** The intersection requested by the GTL profile involves two circles/lines that do not intersect.
NC343	**Coinciding circles** The intersection requested by the GTL profile must be generated by two circles that do not intersect.
NC344	**Coinciding circles/lines/points** The circles/lines/points programmed in the GTL profile are coincident.
NC345	**Points inside circle** Profile error: the programmed point is inside a circle.
NC346	**Parallel lines** Point/circle programming error: the profile lines are parallel.
NC347	**Aligned points** Profile error: the points programmed in the circle definition are on the same line.
NC360	**Too many blocks of movement** The maximum number of blocks of movement allowed inside a profile recalled by a macro rough-shaping (SPA, SPF). has been reached. Check this limit value set in AMP.
NC361	**Profile error** The profile recalled by the macro rough-shaping (SPA, SPF) can not be rough-shaped. In general, only monotonous profiles can be roughed shaped for the rough-shaping axis (which is X or Z always decreasing or always increasing).
NC362	**Undefined work area** Switch off and switch on the control, if error persists, contact the assistance.
NC363	**Axis not congruent with interpolation plane** In the rough-shaping macros (SPA, SPF) the rough-shaping axis must pertain to the interpolation plane, as the axes for which the swarf is defined. Also in the threading macro the thread axis and the return axis must pertain to the interpolation plane
NC364	**Wrong approach to profile** Approach point not allowed for the rough-shaping macro (SPA, SPF). The approach point must always be external to the rough-shaping field in X, for rough-shapings parallel to the X axis, and external to the rough-shaping field in Z, for rough-shapings parallel to the Z axis
NC365	**Interpolation type not allowed** In the profile recalled by the rough-shaping macro (SPA, SPF) only linear or circular blocks of movement are allowed.

Code	Message description and remedial action
NC366	**Aligned points during rough-shaping** During the stage of profile rough-shaping an area that can not be rough-shaped has been reached. Control the consistency of profile and of macro parameters.
NC367	**Profile non consistent with approach** The approach point and the profile development direction don’t allow to continue the rough-shaping.
NC370	**R or B parameters not allowed** In the groove cutting macro is not allowed a connection or initial or final bevel in case the external level has not been programmed.
NC371	**Tool width greater than groove width** Error in the groove cutting macro due to the fact that the tool width is superior to the groove width.
NC372	**Tool width null or not consistent with R or B parameters** Error in the groove cutting macro due to the fact that the tool width is null or inferior to the sum of connections and programmed bevels.
NC373	**Wrong initial position for TGL** The approach position for the groove cutting macro is not consistent with the parameters declared in the block.
NC375	**Missing parameters “a” and/or “b”** If the programmed threading is a non-standard one, in the block must be present also the parameters “a” and “b”.
NC376	**Wrong step for thread** In case of non standard threading it is necessary that the programmed pitch respects the following formula. It must be: $\text{Pitch} \geq 2 * \text{Thread depth} * \text{tg} \frac{\text{thread angle}}{2} * \text{principles number.}$
NC377	**Thread angle greater than 180°** Error in the threading macro due to the thread angle ≥ 180°
NC378	**Null thread length** Error in the threading macro if the thread length along the spindle axis is null.
NC379	**Wrong conical angle** In case of conical threading, the maximum conical admitted is equal to the half of the thread angle.
NC380	**Plane rotation not allowed with thread** It is not allowed to perform a threading cycle if there is active rotation for the interpolation plane.
NC381	**Circular exit not allowed without “r” parameter** Error in the threading macro due to the programming of an output with connection without radius value.

Code	Message description and remedial action
NC401	**HSM Part program not found or open part program error** Possible part program sharing error between the executable modules of the control. Reload the program or contact the customer engineering service.
NC402	**Error reading HSM part program** Part program in execution corrupt Turn the control off and on again or call the customer engineering service
NC403	**HSM configuration file not found or open configuration file error** Setup file not present. Check the presence of the file and the HSM three-letter code that defines the name
NC404	**Syntax error in HSM configuration at line** The specified line contains a syntax error. Check the syntax of the setup three-letter code in the manual.
NC405	**Starting position requested for all HSM defined axis** The first programmed point after the G61 must contain all the axes associated with the HSM setup. Program all missing axes, confirming any positions that do not change.
NC406	**Mandatory HSM param requested into configuration at line** The setup three-letter code set on the specified line requires other parameters Check the syntax of the setup three-letter code in the manual.
NC407	**Mandatory HSM param error** The setup three-letter code set on the specified line does not contain an obligatory parameter Check the syntax of the setup three-letter code in the manual.
NC408	**HSM param at wrong line position** Reserved for future developments. Call the customer engineering service.
NC409	**HSM param not allowed into part program** Reserved for future developments. Call the customer engineering service.
NC410	**Two points are requested to define a segment** There must be at least two points between the G61 and G60 codes. Edit the part program and do not use G61/G60.
NC411	**HSM defined axes not found among the process axis param at line** There must be at least two points between the G61 and G60 codes. Edit the part program and do not use G61/G60.
NC412	**General HSM params must be setted before axis params at line** The setup three-letter code set on the specified line refers to an axis identifier not associated with the process on which the part program is executed. Check the setup three-letter code in the manual or the identifiers of the axes associated with the process.

Code	Message description and remedial action
NC413	**Axis params must be setted after general HSM params at line** In the setup file, the general three-letter codes must be defined first and then the axis setup codes. Check the setup sequence in the manual.
NC414	**HSM needs more configuration params** In the setup file, the general three-letter codes must be defined first and then the axis setup codes. Check the setup procedure in the manual.
NC415	**HSM needs more axis params** Axes with missing setup three-letter codes have been specified in the setup file. Check the setup procedure in the manual.
NC416	**HSM needs more tools params** The tool setup three-letter codes have not been specified in the setup file. Check the setup procedure in the manual.
NC417	**HSM needs more cinematic params** The setup three-letter codes of the axes that refer to the tool and the axes that refer to the clamping of the part have not been specified in the setup file. Check the setup procedure in the manual for the CIN,t and CIN,w three-letter codes.
NC418	**Axis not defined into HSM params at line** An axis not previously defined with the AXI three-letter code has been configured in the setup file on the specified line. Check the setup procedure in the manual.
NC419	**Axis already defined into HSM params at line** A previously defined axis has been defined in the setup file on the specified line. Check the setup procedure in the manual.
NC420	**Too many adding axis (max 3) into HSM params** A maximum of 3 additional axes may be defined in the setup file (axes not belonging to the Cartesian system or rotary) Check the setup procedure in the manual.
NC421	**Too many axis (max 6) into HSM params** A maximum of 6 axes may be defined in the setup file. Check the setup procedure in the manual.
NC422	**Axis type error into HSM params at line** An incorrect axis type or one previously associated with other axes has been defined in the setup file. Check the setup procedure for the AXI three-letter code in the manual.
NC423	**Operative limit reached into HSM part program for axis** The software operating limits have been reached for the specified axis. Check the part program.
NC424	**Virtualization or TCP not allowed with HSM** When the G61 is activated, neither virtualisations nor the TCP must be active. Check the part program.

Code	Message description and remedial action
NC425	**Error reading HSM configuration file** Setup file corrupt. Reload the setup file or call the customer engineering service.
NC426	**HSM not enabled in AMP** The HSM feature has not been enabled in AMP. Enable it.
NC427	**HSM option not allowed by HW key** The HSM option has not been enabled. To use the feature on a machine with more than 3 axes, the option must be enabled using the Product Key Call the customer engineering service.
NC428	**HSM option not loaded** The HSM option has been enabled using the Product Key but has not been loaded onto the NC. Load the option.
NC429	**Illegal param value into HSM** A parameter with an incorrect value (must be positive) has been defined in the setup file on the specified line. Check the setup procedure in the manual
NC430	**Illegal feed value into HSM** The Feed rate value is missing or less than 0. Set a valid feed rate value.
NC431	**Syntax error in HSM** Syntax error in the part program during an HSM machining process. Correct the program and see the programming manual to find out which blocks are allowed between G61 and G60.
NC432	**Illegal use of tangent axes** Only one tangential axis may be present or the tangential axis is being incorrectly used. Check the setup procedure in the manual.
NC433	**Invalid parameter set-up modality** Programming of points type and relative parameter set-up wrong in three-letter PNT code of high speed set-up file.
NC434	**Polynomial programming does not admit parameter set-up requested** Configuration of three-letter PNT code in high speed set-up file for entire polynomial programming is wrong.
NC435	**Nodes must be programmed in increasing mode** Nodes of Bsplines programmed as inputs must be sorted in increasing order.
NC436	**Node programming requested** Number of nodes programmed is insufficient: for Bspline inputs, the number of nodes must be the same as the number of control points, plus the degree of Bspline + 1.

Code	Message description and remedial action
NC437	**Final point of previous Bspline must be confirmed** Programmed Bsplines must be continuous, i.e., last point in a Bspline must be the same as first point in the next.
NC438	**Control points for correct definition of Bspline missing** Minimum number of control points, for Bspline input, must be: (degree of Bspline + 1) *2.
NC439	**Programmed polynomials lack continuity** Programmed polynomials must be continuous, i.e., last point in a polynomial must be the same as first point in the next.
NC440	**ROT type IJK vector invalid when Tangent axes computation is requested** When working with a tangent axis, the ijk vector must not be ROT type in three-letter VER code of high-speed set-up file.
NC441	**ROT type IJK vector invalid when (TOD) parameters are used** In three-letter VER code of high-speed set-up file, ijk vector cannot be set as ROT when, for instance, drive chain includes fewer than two rotating axes.
NC442	**PRG type IJK vector invalid when type AXI/CLP points are programmed** When programming type (PNT, AXI/CLP... or (PNT, AXI/CCP... points, ijk vectors cannot be used. Use one of the following instead: (VER, REL/ROT, ...
NC443	**REL type MNO vector invalid when type AXI/CCP points are programmed** When programming type (PNT, AXI/CXP... or (PNT, CLP, CCP... points, mno vectors cannot be used. Use: (VER..., PRG, ...
NC444	**Axis shared with PLUS environment** The axis you are trying to move, or on which you wish to perform a virtualisation, has been previous acquired by the logic through the AX_SHARE function.
NC445	**Machine unit of measurement not congruent in HSM** This error occurs during the execution of a part program optimised with Path Optimizer when the unit of measurement specified in the setup file is not congruent with the unit of measurement of the machine.
NC446	**mno/uvw programming wrong in HSM file** This error occurs when both the mno/uvw vectors are defined, programmed or calculated in the VER three-letter code of the high speed setup file.
NC447	**uvw programming not compatible with toroidal tool** The uvw compensation factors are not supported with toroidal tools.
NC448	**Invalid corner for the determination of uvw** This error is generated when you try to determine the uvw compensation factors for a point where an overelongation has occurred. That is to say, tilting by 180° relative to the point of contact on the tool occurs.
NC456	**ijk versor is null** The ijk module versor programmed is null

Code	Message description and remedial action
NC457	**mno versor is null**
NC458	**pqd versor is null**

END OF APPENDIX

ERROR MANAGEMENT

GENERAL

This Appendix discusses how the operator can manage errors in order to prevent machining process interruptions.

The following system variables permit to configure the error management mode:

SYSTEM VARIABLE	FUNCTION
ERR	Enables error management from part program
STE	Is a read only system variable that contains the error generated by the previous command when automatic error management is active.

ERR - Enable/disables error management from part program

ERR is a system variable that permits to select how to manage the error.

Syntax

ERR = *value*

---

where:

value	It may be 0 or 1 and can be expressed with a number or a local or system variable. ERR=0 (default) disables error management from program is disabled. Errors are displayed and can interrupt program execution. ERR=1 enables error management by the part program.

Characteristics:

The tables that follow list a series of programming errors that can be managed from program. For each error with provide the error code (value written in the STE read-only variable) and a short description.

In addition, the table shows the NCxx that is displayed when ERR=0. For the complete list of NCxx codes refer to Appendix B.

Probing cycle errors

Command	STE value	Description	Error code displayed if ERR=0
G71,G72,G73	0	probing cycle finished regularly	----------
	1	probing has not taken place	NC098
	2	probe has not been released	NC099
	3	probe parameters not specified	NC007
	4	probing has taken place with rapid approach	NC115

Example:
Presence of the part is verified with both methods:

```
(DPP,30,15,500)
G0 Z50
X80 Y100
ERR = 1
G72  Z0  E1
(GTO,END,STE = 1)
ERR = 0
.......................................
.......................................
"END" (DIS, "PART NOT PRESENT")
M...
.......................................
```

IMPORTANT

Since G71,G72,G73 modify the value written in the STE variable when ERR=1, we recommend that you use STE immediately after giving a command that might alter its contents.

Shared axes errors

Command	STE value	Description	Error code displayed if ERR=0
GTA	0	execution without errors	**----------**
	10	axis ID not configured	NC135
	11	axis ID belongs to logic	NC136
	12	axis ID belongs in another process	NC137
	13	axis ID associated to spindle axis	NC139

Example:

```
;. . . . .
;. . . . .
ERR = 1
;. . . . .
"RETRY"
(GTA,X1,Y2,Z5)       ;Requests acquisition of ID's  1,2,5
(GTO,NEXT,STE<>12)          ;Axis error test busy
(DLY,0.5)                   ;If the axis is busy the program waits until it is released
(GTO,RETRY)
"NEXT"
(GTO,ERROR,STE<>0);If other error occurs the cycle will be aborted
G1 X10 Y10 F1000            ;After the axes have been acquired the move is executed
Z50
;. . . . .
;. . . . .
;. . . . .
"ERROR"
;. . . . .                  ;Error management
```

IMPORTANT

- Error management from part program also provides an easy way of synchronizing two processes when one of them must wait for one or more axes to be available before executing a given process.
- Since GTA modifies the value written in the STE variable when ERR=1, we recommend that you use STE immediately after giving the command that might change its contents.

Errors in multiprocess management

Command:	**Description**	**Error code displayed if ERR=0**
EXE,ECM	0 execution without errors	------
	1 EXE or ECM command failed	NC227
	2 Process type wrong	NC221
	3 Process number wrong	NC222

Example:
Verify the correct execution of the synchronous command on process 2 using the error management method:

```
ERR = 1
(ECM,”G1 X100 F1000”,P2,S)
(GTO, ERROR,STE <> 0)
ERR = 0
.......................................
.......................................
;. . . . .
"ERROR"
;. . . . .                              ; Management of error cases
```

END OF APPENDIX