topCAD™ Syntax Reference

Pantomat

www.pantomat.com

First Edition June, 1993
Second Edition March, 2018

CONTENTS

About This Manual

This manual, **topCAD Syntax Reference**, describes the syntax and programming tools that make up the topCAD Programming Language. Refer to this manual for the syntax required to type commands from the keyboard, write macros, create new drawing methods; or create user icons, user menus or a video tablet.

In this manual you will find:

- ☐ The functional description of all topCAD commands
- ☐ The syntactical description of all topCAD commands
- ☐ The description of all programming facilities you need to create and use macros and user defined commands

The topCAD documentation set includes two other manuals, the **topCAD Installation & Demo Guide** and the **topCAD User's Guide.**

topCAD Installation &Demo Guide introduces topCAD, describes how to install it, and also gives a guided tour of the program. *topCAD User's Guide* describes the working environment, tools and commands menu by menu. Also, it contains appendices, a glossary and an index.

These manuals assume that you are familiar with Macintosh®. If you are new to Macintosh, please review your Macintosh owner's guides.

Organization of This Manual

Chapter 1, *Introduction,* describes the purpose of topCAD's programming language along with the general syntactical rules.

Chapter 2, *Detailed Description of Commands,* describes the functions, syntax and use of topCAD's commands in detail.

Chapter 3, *Programming Tools,* describes the other programming facilities you need in order to write macros: such as expressions, functions, variables, assignments, control statements; and how macros can be supplied with a help function in topCAD's programming language. Here you will find a list with a bunch of C routines implemented by Graphisoft for calling external procedures from topCAD.

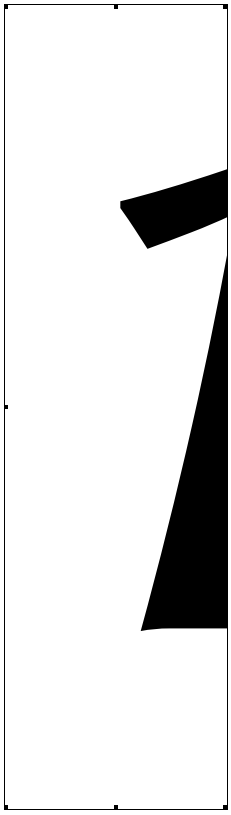
Chapter 4, *Macro /External Procedure Examples,* gives several examples, with varying degrees of complexity, of writing macros and the use of external procedures written in high-level programming languages such as C, Pascal and FORTRAN. Explanation and figures are also given with the examples.

Appendices lists keywords, commands' syntax in alphabetic order, functions, system variables, font IDs and Apple events in topCAD.

A ***Command Index*** lists the commands by keywords.

C H A P T E R

Introduction



Using topCAD Commands

In this manual you can find the functional and syntactical description of each basic topCAD command, followed by an example if necessary.

Syntactically correct topCAD commands can be used:

One by One

By typing in the command window

- The keyword of the command
- Other keyword(s), if any
- Expression(s) and/or parameter value(s) necessary to execute the command
- ENTER

This is equivalent to command execution by the graphic tools (icons, menus) but

- You can use all topCAD commands
- You should know and apply the rules of syntax

In a String of Commands

By editing a series of commands according to the syntactical rules. This string of commands is called *compound command* and provides a complex function according to the needs of your application environment.

What is This Command Description for?

Working with topCAD you can execute most commands one-by-one through the graphical user interface tools, icons and menus; consequently, you don't need to know the command syntax.

But, as mentioned above, topCAD provides the facility to write compound commands, i.e., arbitrary series of basic commands to realize any complex function you need.

These compound commands can be used as macros; you can run the macro as a program, and all commands in it will be executed.

But, you can use compound commands through the graphical user interface as well, by assigning them to your own tools, menus, icons, or video tablet icons. Since this customization can improve the efficiency of you work dramatically, hopefully this manual will help you get the most from topCAD's capabilities and will contribute to success in your work.

About Programming in topCAD

Using topCAD's programming language you can write compound commands, which may include:

- □ Arbitrary series of all basic topCAD commands
- □ Any number of previously defined macros
- □ Functions
- □ Control structures and other facilities

according to the programming rules described in chapter *General Programming Tools*.

Any compound command can be defined and used as a

- □ Custom tool command
- □ Custom menu command
- □ User icon command
- □ Video tablet command
- □ Macro

All these kinds of compound command are of similar structure but their use is different.

You can assign compound commands

- □ User defined tool options, by the *Customization Tools...* command. E.g.: a new circle definition can be made as a tool command
- □ Custom menu items by the *Customization Menus...* command. (Thereafter it can be used as a user menu command.)
- □ User defined user icons by the *Customization User Icons...* command. (You can change any user icon and the command(s) assigned to it.)
- □ User defined video tablet icons by the *Customization Video Tablet...* command (you can create a palette of user defined tools on the screen)
- □ Macros defined by the *New Macro* command

It is advisable to define the compound commands performing frequently used operations or drawing methods as custom tool, user menu, video tablet or icon commands. Their use is simple and quick; however, their number is limited. (See the *Customization* command in the File menu in topCAD's User's Guide.)

The compound commands creating objects (symbol definitions, etc.) should be defined as macros, for the number of macros is unlimited.

Macros

Macros are series of topCAD commands and earlier defined macros and user procedure calls. They are not executed directly from a menu but by a special command (see below).

To define a macro

Choose the *Open Macro New* command from the File menu. After giving a name you can assemble the macro from the basic topCAD commands and existing macros. There is no limit to nesting macros. Having finished the macro definition, you can put it into the Macro folder (default) or another folder.

To use a macro

Choose the *Execute Macro* command from the File menu. After selecting and opening the macro it will be executed.

To edit a macro

Choose the *Open Macro* command from the File menu. After selecting and opening the macro it can be edited. (See section *The File menu* in chapter *The Menu Commands* in the User's Guide for details).

A simple way of writing macros

A simple way of writing macros is to record steps while using topCAD.

The steps to follow when you create a macro by recording:

- Start recording by checking *Journal file* in the Preferences Miscellaneous dialog box in the Attributes menu.
- Execute the steps you want to record.
- Stop recording by checking *Journal file* off in the Preferences Miscellaneous dialog box.
- The recorded file named *topJournal1.1*, *topJournal2.1*,...,*topJournal1.2*, *topJournal2.2*,... can be opened by the *Open Macro* command in the File menu.

IMPORTANT: The Journal files are located in the List folder by default. They are deleted while quitting topCAD. If you want to use a Journal file at a later time follow the steps below.

- Choose *Open Macro* from the File menu.
- Check the *List* option in the Open dialog box.
- Open *topJournaln .m* in the List folder.
- Click on the *Save As* button in the opening text editor box.
- Enter a name for the macro to be saved and click on *OK*.

The Command Line

This description contains the instructions in the format they should be entered. In one command string there can be an arbitrary number of commands. Object creating or modifying commands are main commands, for example. Other commands (*subcommands*) can be placed before or after the main command or inserted between any two syntactical elements. A subcommand can also be broken by a new subcommand.

The simple structure of commands (either main or subcommands) is:

/KEYWORD < [/MODIFIER < par >] > < par >

A command may consist of a command keyword followed by the parameters required by the keyword. Between the keyword and its parameters one or more keyword modifiers, each followed by its own parameters, may be inserted or omitted, as needed.

E.g.: The syntax of the /ZOOM command is:

/ZOOM [IN | OUT] coord1 coord2

In the command line

/ZOOM /IN 100 100 200 300

ZOOM:	Keyword
IN:	Modifier
100 100:	Corner of the zoom rectangle
200 300:	Opposite corner of the zoom rectangle

Keyword Defines the command. The keyword must be preceded by a slash "/".

Modifier Identifies a particular function of the command. The modifier must also be preceded by a slash "/".

Parameter Numerical or textual value or expression.

Textual parameters may be among the parameters of a command. The text must appear within quotation marks ("text").

A parameter for a command may come from the command line, as well as from the user.

Prompt Each command waiting for parameters has its individual *prompt* which immediately appears in the *message line* when the program recognizes the keyword. This keyword prompt informs the user about the necessary parameters or qualifier keywords.

In general, topCAD uses the standard prompts belonging to each command. The standard prompt can be replaced.

E.g.: /ZOOM /IN '#' '#' #Displays the standard prompts

```
/ZOOM /IN       'Specify a corner of the zoom rectangle'
                'Specify the opposite corner of the zoom rectangle'
```

Syntax Rules

1. The slash “/” preceding a string instructs topCAD to interpret the following string as a keyword or modifier.

2. A semicolon (**;**) is equivalent to the /ENTER command.

   E.g.: the /CIRCLE /CPOINT 100 100 200 200 /ENTER
   and the /CIRCLE /CPOINT 100 100 200 200 ;

   command lines have exactly the same effect: both draw a circle with the center point (100; 100) passing through the point (200; 200) and quit the command.

3. The syntactical elements of the command lines should be separated by one or more spaces “ “ or commas “,”.

   E.g.: /CIRCLE 140 230 ;
   /CIRCLE 140,230 ;
   /CIRCLE 140, 230 /ENTER

   All the commands above have the same effect.

4. An expression may contain any number of spaces.

   E.g.: (a+2*b) and (a +2* b),
   (i <128) and (i < 128)

   are equivalent.

5. The following syntactical elements must be given between delimiters:

   □ *Prompt references* between ' ' characters

   A prompt reference should be used when a parameter is to be expected from the user rather than coming from the command line.

   Both the # for the standard prompt and the user defined prompt must be given between ' ' delimiters.

   E.g.: /REPEAT '#' /DUPLICATE /SELECTION
   /REPEAT 'Enter the repeat factor:'/DUPLICATE /SELECTION

   □ *Text* and *File names* between " " characters.

   E.g.: "This is a text"

   □ Arithmetical and logical *expressions* between parentheses ( ).

   E.g.: (13 + .3*var1/c)
   (a+2*b)
   (i<128)

- *Format list* between " " characters.

  E.g.: /LIST "%f %d" (a1) (a2) ;

  where a1 is a floating point-type, a2 is an integer-type variable.

- *Indexes* of array elements between [ ] characters.

6. No delimiter is needed

   - In the /USE command

     E.g.: /USE var2

   - In any variable definition (except the /VTEXT command)

     E.g.: /VINT var2 876

     The variable var2 will have the value of 876.

   - In a text variable definition for the variable name. Note that the value itself must be enclosed in " " delimiters.

     E.g.: /VTEXT var5 "This is a text"

     The variable var5 will have the value of This is a text.

7. Numerical values can be given between parentheses ( ) but it is not obligatory.

   E.g.: Both forms (125) and 125 are correct.

8. There is no difference between the upper/lower case characters in keywords, variable names, symbol names and filenames.

   E.g.: the variables named aBC, ABC or abc are the same,
   the /circle and the /CIRCLE keywords are the same.

9. In text the upper/lower case characters are different.

   E.g.: Define a *Preferred Text*:

   /ATEXT "Abcde"

   Locate it:

   /TEXT 100 150 ;

   The string is located keeping the upper/lower case character of the letters.

10. With keywords, you need only type enough of a keyword to make it unique; the first matching keyword will be used. However, we suggest that you enter all the keywords to avoid ambiguity.

    E.g.: /PDELETE 100 100 ; and /PDE 100 100 ;

    are identical because PDE unambiguously identifies the PDELETE command, but you cannot enter

    /POLY instead of /POLYLINE

    because there is another command /POLYGON starting with the same letters.

11. Uses of the # character in macros:

    Comment:

    A # can be placed anywhere in a command line; the part of the text between the # and the end of the line will be considered as a comment and will not be interpreted. Consequently, if the first character in a line is a #, the whole line will be treated as a comment.

    E.g.: /CIRCLE 130 140 ; #This is a command drawing a
    #circle of the Preferred Radius

    The circle will be drawn and the comment reminds you of the function of the command.

    Standard prompt:

    Entering '#' following the keyword of a command leads to the standard prompt of the command appearing in the command window.

    Using files in macros:

    Defining a variable, it is possible to choose its value from a file.

    E.g.: /VINT var3 '#Dfilenums'

    You will see a dialog box displaying the numbers in the file named *Filenums*. The value you select will be assigned to the variable var3.

12. File-, symbol- and variable names:
    The name must start with a letter, followed by arbitrary characters.
    - □ filename: max. 120 characters (this is the maximal length of the total file definition including volume and/or folder names),
    - □ symbol name: max. 50 characters,
    - □ variable name: max. 12 characters.

A variable reference may consist of:

- □ the name of the variable,
- □ one or more indexes (in case of arrays), and
- □ a qualifier (in case of coordinate and transformation variables). (For qualifiers refer to *Variables* in chapter Programming Tools.)

E.g.: shift.11
Q[2][3].r
/list "%f" (pos[0][1].y) ;

13. Wildcards in filenames and symbol names:

    % any single character,
    * any number of characters following the position of this asterisk.

14. Entering values in imperial mode.

    The unit must be given as
    " for inch
    ' for foot
    and value and unit together must be placed between parentheses.

    E.g.: 4 1/4 inch: (4 1/4")
    2 feet 2 1/8 inch: (2'-2 1/8")

15. You should give an ENTER to quit the “cyclic” commands since they restart after execution.

    E.g.: as a result of the

    /CIRCLE 140 230

    command, topCAD draws a circle with the center point at (140; 230) and prompts for the center point of the next circle.

    Having received the

    /CIRCLE 140 230 /ENTER or the

    /CIRCLE 140 230 ;

    command, topCAD draws the circle, quits the command and tries to execute the next command (if any).

16. The length of command lines is unlimited.

Conventions in This Manual

Typography Commands appear in three different ways throughout this manual in the syntactical description of topCAD programming language.

ZOOM [IN] coord1 coord2 This is the precise syntactical description of the command. Punctuation shows which elements of the command line are alternatives, optional, or can be repeated as many times as desired. In the command syntax

UPPERCASE words are keywords or modifiers, e.g.: ZOOM [IN]

lowercase words are parameters, e.g.: coord1 coord2

/ZOOM /IN 100 100 200 200 This is an example of the command. The command is to be entered as it is, though lowercase keywords are also accepted.

```
/ZOOM /IN 100 100 200 200
```

The commands are shown this way when command lines appear in a list of a complete file. The file is shipped together with the topCAD system, and put to disk at installation.

Note the above rules are used for easy reading; while writing macros you can use either upper or lowercase characters.

Punctuation These syntactical elements appear only in the syntactical description of topCAD commands. They should not appear in the command string, and only show which elements of the command line are alternatives, optional, or can be repeated as many times as desired.

...|... Vertical bars separate alternative elements. One and only one of those elements can be put into the command line.

{...|... ...} Braces enclose the alternatives. One of the listed elements must be selected.

[...|... ...] Brackets enclose one or more optional items. For example:
/LAYER { /ACTIVE | /ON | /VISIBLE | /OFF } [/ONLY] set
This command sets the state of each layer given in set depending on the keyword that follows /LAYER. One of the four layer states must be given while the keyword /ONLY is optional.

<...> Angle brackets enclose item(s) which can be repeated any number of times. An Enter to terminate the repetition is always required and involved in the command, though not explicitly shown. For example:
/POINT < coord > must appear like this: /POINT 10 20 10 50 ;

Parameters The syntactical elements below appear in the syntactical description of topCAD commands as parameters. Where they appear the program expects an appropriate value, pair of values, or set of values – depending on the parameter. In general, anything that results in a value of the appropriate type can be entered in the command line where it expects a parameter; such as numerical values, expressions, or commands.

value — A single value of any type.

E.g.: 123
(i+2*/PI)
topAlpha

valint — An integer type value.

E.g.: (i*5)
/SIZE /RECALL

valfloat — A floating-point type value.

E.g.: /PI
12.73
/USE fvar

coord — A pair of coordinates.

E.g.: /CRTCENTER
100 130
/RECALL

length — A length value.

E.g.: (12")
/LIKE 76 151

angle — An angle value.

E.g.: 45
/APARALLEL 100 100

text — Textual value.

E.g.: textwithoutbreak
"text with break"
/DATE

file — Text containing the specification of a file.

E.g.: "List:Myfile"
'#FM'

symb — Text containing a symbol name.

E.g.: constrain
symbol*

var — Text containing a variable name.

E.g.: textvar, B[5][2]
tran.11

tran — A transformation.

E.g.: toptrans
/XLATE 0 0 100 100

num — A single arithmetic expression.

E.g.: (i+n[2][3].x*/PI)

set — A set of natural numbers in the range 0 – 255

- The keyword /NO followed by a numerical expression
- A pair of numerical expressions separated by an underline
- The keyword /ALL

E.g.: /ALL
1_7

single — Selection of a single object.

E.g.: 100 100
/RECALL

group — Selection of a group of objects.

E.g.: 100 100 100 200
/SLTYPE /ALL /EXCEPT 0_2

Illustrations Illustrations, when necessary, are given to explain the commands. They are designed to make things easier to understand rather than represent exactly what appears on your monitor. The illustrations use the following line type conventions:

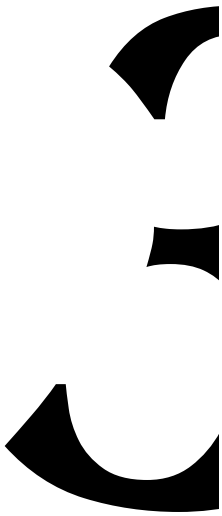
Existing objects ——— – ——— – ——— – ———

Auxiliary lines – – – – – – – – – – – – – – – – – –

Resulting objects ———————————————

C H A P T E R

Draft Managing Commands



Control Commands

NEW { YES | NO }

Opens a new, empty drawing. The effect of the command depends on the number of currently open drawings.

If there is no or only one drawing open when the command is executed

- Opens an empty *Untitled* drawing

If there are two drawings open when the command is executed

- Closes all detail windows of the currently active drawing
- Clears the main window (deletes all objects in the current drawing)
- Sets the drawing name to *Untitled*

All changes since you last saved your current drawing will be lost. There is no way to revoke the lost objects by the /UNDO command.
The command needs a confirmation either in the command line or from the keyboard. If the answer is NO, the command has no effect.

E.g.: /NEW /LINE /SINGLE 100 100 200 200 ;

topCAD prompts for confirmation.
Entering YES executes the command.
Entering NO cancels the command.

Whatever the answer, the rest of the command line following /NEW is not executed.

/NEW /YES

topCAD executes the command.

/NEW /YES /LINE /SINGLE 100 100 200 200 ;

topCAD closes the current drawing, opens a same size Untitled one with a straight line in it.

QUIT { YES | NO }

Quits topCAD. The command needs a confirmation.
If the answer is NO, the command has no effect.

E.g.: /QUIT

topCAD prompts for confirmation.
Entering YES executes the command.
Entering NO cancels the command.

/QUIT /YES

topCAD executes the command.

ENTER

Finishes the execution of the current activity and returns

- ☐ to the nesting command, if any, or
- ☐ to the Command prompt. The ";" is equivalent to /ENTER.

E.g.: the commands

/CARC (123) (121) /ENTER
/CARC (123) (121) ENTER
/CARC 123 121 ;

are equivalent. See section *Terminating Commands* in chapter *Commands in General* in User's Guide for details.

ABORT

Terminates the current activity without execution and returns

- ☐ to the nesting command, if any, or
- ☐ to the Command prompt. See section *Terminating Commands* in chapter *Commands in General* in User's Guide for details.

CANCEL

Terminates all activities without execution and returns to the Command prompt. topCAD will wait for a new command. See section *Terminating Commands* in chapter *Commands in General* in User's Guide for details.

DUMMY

It has no function in itself but, as it is a main command, any following change made to the *Preferred values* or *Attributes* will be TEMPORARY. It is very useful to start macros by this command.

WAIT sec

The program waits for sec seconds.

E.g.: /WAIT 17

Before the execution of the next command topCAD waits for 17 seconds.

PAUSE

The program waits for a click.

View Commands

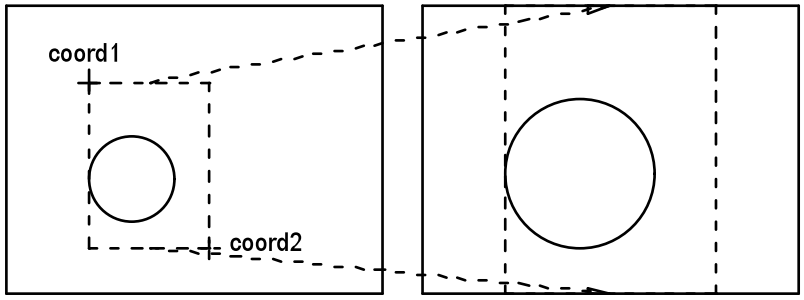
The View commands affect only the display of the current drawing or a part of it in the drawing window. No coordinates or sizes change in the drawing while using the following commands.

ZOOM [IN] coord1 coord2

The chosen area will fit the active drawing window, retaining the original vertical/horizontal ratio.

The center of the chosen area will be located at that of the drawing window.

coord1: a corner of the chosen rectangular area;
coord2: the opposite corner of that area.



Before After

E.g.: /ZOOM /IN (100) (100) (200) (200)

The contents of the rectangular area defined by the points (100; 100) and (200; 200) as its corners will replace the contents of the active drawing window.

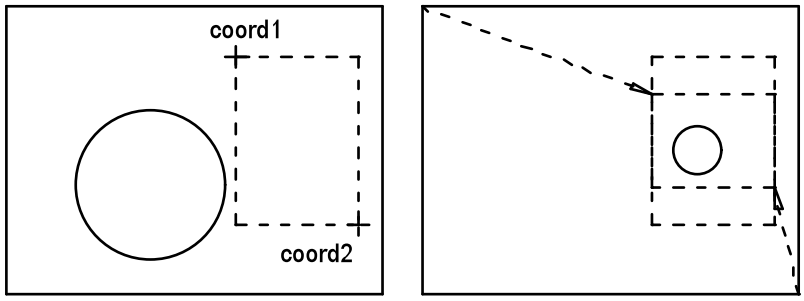
/ZOOM /IN /CRTCENTER /BOTTOMLEFT

The contents of the lower-left quarter of the active drawing window will be enlarged to full window size.

ZOOM OUT coord1 coord2

The contents of the active drawing window will be compressed into the chosen area, retaining the original vertical/horizontal ratio.
The center of the drawing window will be projected to that of the chosen area.

coord1: one of the corners of the chosen area;
coord2: the opposite corner of that area.



Before After

E.g.: /ZOOM /OUT (100) (100) (200) (200)

The whole contents of the active drawing window will be compressed into the rectangular area defined by the points (100; 100) and (200; 200) as its corners.

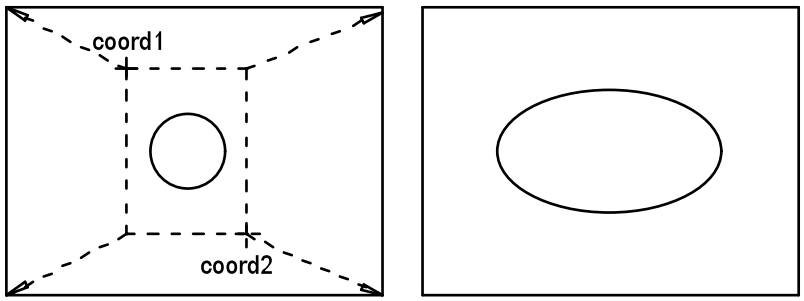
/ZOOM /OUT /CRTCENTER /BOTTOMLEFT

The whole contents of the active drawing window will be compressed into its lower-left quarter.

WINDOW coord1 coord2

The chosen rectangular area completely fits the active drawing window.
The vertical/horizontal ratio will, in general, change!

coord1: one of the corners of the chosen window;
coord2: the opposite corner of the chosen window.



Before After

E.g.: /WINDOW 100 100 180 200

The contents of the window defined by the (100; 100), (180; 200) corners will fill the whole active drawing window.

AUTOSCALE

Scales the drawing so as to display all visible objects and optimally fill the active drawing window. The vertical/horizontal ratio does not change.

SCALE num

Enlarges or reduces the drawing through the active drawing window, retaining the position of its center. The vertical/horizontal ratio does not change.

num: the scaling factor
num>1 The drawing is enlarged
num<1 The drawing is reduced

E.g.: /SCALE 2.7

The drawing will be enlarged by 2.7.

QSCALE { LTYPE | LWIDTH } { ON | OFF }

Determines whether line type or line width should also be scaled along with the drawing.

E.g.: /QSCALE /LWIDTH /OFF

Line width of lines will not be affected by any SCALE command which follow.

RATIO { num | ENTER }

The vertical scaling will be multiplied by num, retaining the original center point (the vertical dimensions change, the horizontal dimensions remain unchanged), or the command restores the original dimensions of the drawing.

num: the vertical scaling factor;
num<1: the drawing is reduced vertically by num
num>1: the drawing is magnified vertically by num
ENTER: the vertical dimensions revert to the original values.

E.g.: /RATIO 2

The drawing will be vertically enlarged by a factor of 2.

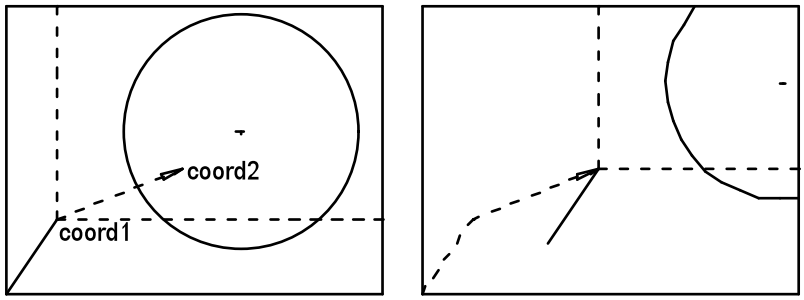
/RATIO ;

The Y/X ratio will be reset to the original value.

PAN coord1 coord2

Moves the drawing window over the drawing in such a way that the point coord1 of the command window is relocated to the point coord2. The vertical/horizontal ratio remains unchanged.

coord1: the startpoint of the shift vector;
coord2: the endpoint of the shift vector.



Before After

E.g.: /PAN 50 80 150 80

The drawing will be shifted in the drawing window to the right by 100 drawing units set by *Length Unit* in the *New* dialog box on the File menu or in the*Preferences Units* dialog box on the Attributes menu.

PREVIOUS num

Displays the (current - num)th view in the active drawing window. The contents of each window are stored separately, in 20 entry–long ring buffers.

FOLLOWING num

Displays the (current + num)th state of the active drawing window. The contents of each window are stored separately, in 20 entry–long ring buffers.

PSCALE [SET num] coord

Displays the drawing according to the paper scale factor you set.

SET: Keyword to set the paper scale factor for displaying the drawing

num: The scale factor; e.g. a value of 10 means that a circle with a radius of 10 mms appears **on the screen** as a 1 mm radius circle.

coord: A point of the drawing which will be displayed at the center of the active drawing window.

E.g.: /PSCALE /SET 2 100 100 ;

sets the scale factor to 2; the coordinates of the point in the middle of the screen will be 100,100.

REDRAW

Redraws the contents of the active drawing window.

SHOW group

In the active drawing window the program restricts displaying to the selected group of objects. Note that objects drawn after executing the command appear in this window, too. The effect of the command ceases if an implicit /REDRAW (/AUTOSCALE, /SCALE, ...) is specified.

RGB { DEFAULT | < num red green blue > }

Assigns a red, green, blue vector to the num logical color number or resets the default logical color table vectors. The values may vary from 0 to 65535.

DEFAULT: keyword to reset the values to default;
num: the code of a color whose red-green-blue vector is to be redefined;
red, green, blue: value of the red, green and blue component of the color (num).

E.g.: /RGB /DEFAULT;

resets the red-green-blue vectors for all the 256 logical color numbers to the default values.

/RGB /DEFAULT 77 10000 20000 30000;

assigns default to all logical colors except the logical color number 77 which will be assigned the values red=10000, green=20000, blue=30000.

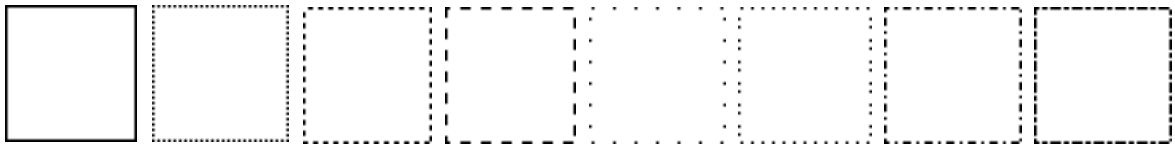
/RGB 56 34000 20000 12345;

assigns the red=34000, green=20000 and blue=12345 values to the logical color number 56.

GRID { ON | OFF | num1 num2 num3 num4 num5 length1 length2 }

Displays, clears or defines grids to all windows of the current drawing. As many as four grids can be displayed at a time.

ON: displays the grid;
OFF: hides the grid;
num1: the serial number of the grid (1-4);
num2: the type of the grid:



1 2 3 4 5 6 7 8

num3: the rarefaction factor (every num3th grid line will be displayed);
num4: the logical color (0-255);
num5: that maximum number of lines above which the grid is not displayed, since it would become too dense;

length1: the grid spacing in x direction;
length2: the grid spacing in y direction.

The default values are:

num1	num2	num3	num4	num5	length1	length2
1	6	1	1	50	1	1
2	5	1	2	50	10	10
3	4	1	3	50	100	100
4	1	1	4	50	1000	1000

E.g.: /GRID 4 2 3 100 20 15 25;

sets the parameters of grid 4 as follows:

- ☐ dotted line type (2),
- ☐ every 3rd grid line will be displayed,
- ☐ the grid lines will be drawn with the color of code 100,
- ☐ the horizontal spacing between two neighboring gridlines is 15,
- ☐ the vertical spacing between two neighboring gridlines is 25.

CLOSE { YES | NO }

topCAD closes the active drawing, which means it closes all windows of the active drawing and the drawing file too. The command needs a confirmation. If the answer is NO, the command has no effect.

LAYER NAME < num name >

Defines new layers with the given names, or eliminates empty layers.

num: layer number for numerical identification;
name: the name of the layer.

E.g.: /LAYER /NAME 2 "dimensions" 3 "hatches" ;

Creates layer 2 and 3 named “dimensions” and “hatches” respectively.

/LAYER /NAME 2 "" ;

Eliminates layer 2 if it is empty, or
resets the layer name to “Untitled” if the layer has objects in it.

LAYER { ACTIVE | ON | VISIBLE | OFF } [ONLY] set

Sets the layer(s) specified by set to one of the following states:

ACTIVE Results in an *Active* layer which behaves like this:

- ☐ the new objects will be assigned to this layer (input layer),
- ☐ the objects assigned to this layer can be modified,
- ☐ the objects in this layer can be seen in the drawing window,
- ☐ the objects in this layer can be referenced,
- ☐ the currently ACTIVE layer will be set to the state ON.

ON Results in *Modify Only* layers which behave like this:

- ☐ the objects assigned to these layers can be modified (output layer),
- ☐ the objects in these layers can be seen in the drawing window,
- ☐ the objects in these layers can be referenced,
- ☐ if the ONLY keyword is present, the currently ON layers which are not in the selection will be set to the state VISIBLE.

VISIBLE Results in *Visible* layers which behave like this:

- ☐ the objects in these layers can be seen in the drawing window,
- ☐ the objects in these layers can be referenced,
- ☐ if the ONLY keyword is present, the currently VISIBLE layers which are not in the selection will be set to the state OFF.

OFF Results in *Invisible* layers which behave like this:

- ☐ the objects in these layers will disappear from the drawing window,
- ☐ if the ONLY keyword is present, the currently OFF layers which are not in the selection will be set to the state VISIBLE.

set: the selected set of layers. Note that if the /ACTIVE keyword is given the set can contain only one element.

E.g.: /LAYER /ON 240,245,250 ; and press the ENTER key

(from the command window) or

/LAYER /ON 240,245,250 ; ;

(from a macro) the layers 240, 245 and 250 will be ON, the other layers will not be affected.

/LAYER /ACTIVE 240 ;

the layer 240 will be ACTIVE, the current ACTIVE layer will be ON.

/LAYER /VISIBLE /ONLY 240_245 ;

the layers 240, 241, 242, 243, 244, 245 remain VISIBLE and all the other VISIBLE layers will be OFF.

Object Creating Commands

With the help of these commands a variety of objects can be created (e.g. a circle with a given radius, tangential to two other objects). The order of the parameters in a sequence is always definitive.

Point Defining Commands

POINT < coord >

Locates a point at the specified position.

coord: the position of the point.

E.g.: /POINT 100 130 ;

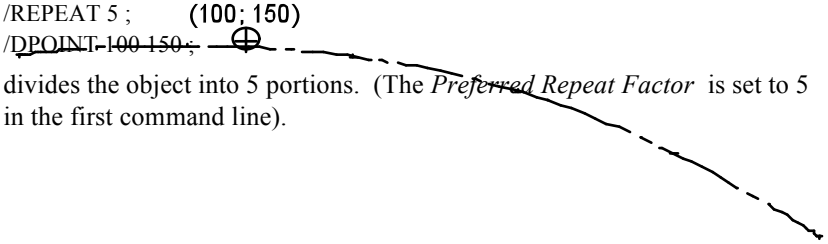
a point will be created at the position (100; 130).

DPOINT < single >

Divides the selected object along its length (or perimeter) into *n* equal portions (where *n* is the *Preferred Repeat Factor*) and creates a point at the endpoints of each portion.

single: a point near the object to be divided.

E.g.: /REPEAT 5 ;
/DPOINT 100 150 ;



divides the object into 5 portions. (The *Preferred Repeat Factor* is set to 5 in the first command line).

Line Defining Commands

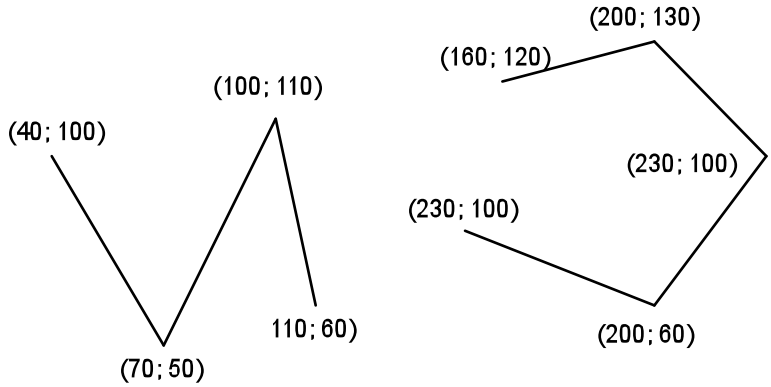
LINE < < coord > >

Defines a series of lines connected at the endpoints.

coord: an endpoint of a line. Enter completes a series of connected lines and repeats the command, i.e., starts another series of lines. Giving another Enter completes the command.

E.g.: /LINE 40 100 70 50 100 110 110 60 ; 150 80 200 60 230 100 200 130 160 120 ; ;

The first ENTER (;) terminates the first series of lines.
The second ENTER terminates the second series of lines.
The third ENTER (right after the second one) terminates the command.

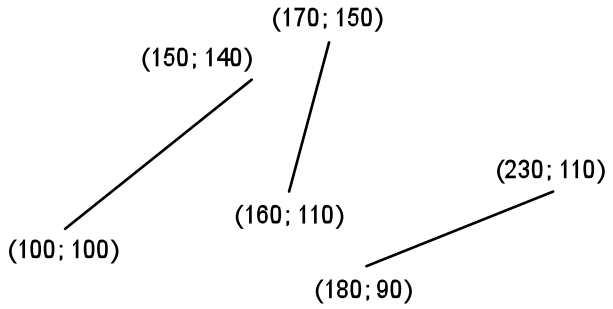


LINE SINGLE < coord1 coord2 >

Defines single lines by their endpoints.

coord1: the startpoint of a single line;
coord2: the endpoint of a single line.

E.g.: /LINE /SINGLE 100 100 150 140 160 110 170 150 180 90 230 110 ;



LINE PERPENDICULAR < coord single >

Defines a line starting from the selected point and perpendicular to a selected object.

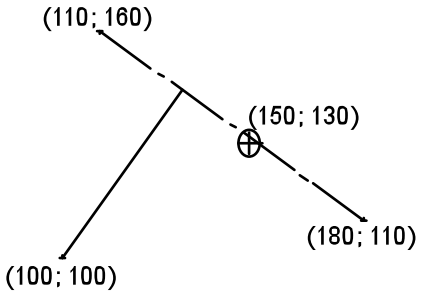
coord: the endpoint of the line;
single: a point near the object the line will be perpendicular to.

E.g.: /LINE /SINGLE 110 160 180 110 ;

Draws a line connecting the points (110; 160) and (180; 110).

/LINE /PERPENDICULAR 100 100 150 130 ;

Draws a line starting from the point (100; 100) perpendicular to the previously defined line.

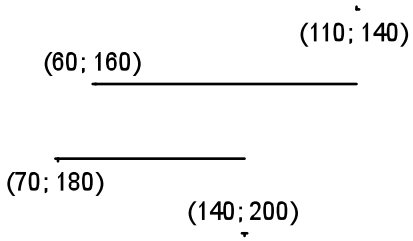


LINE HORIZONTAL < coord1 coord2 >

Defines single horizontal lines.

coord1: the startpoint of a horizontal line;
coord2: the x coordinate defines the endpoint of the line. The y coordinate will be ignored.

E.g.: /LINE /HORIZONTAL 60 160 110 140 70 180 140 200 ;

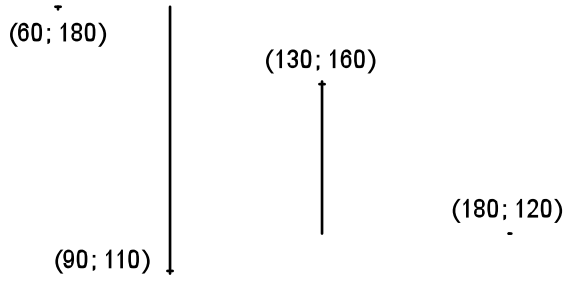


LINE VERTICAL < coord1 coord2 >

Defines single vertical lines.

coord1: the startpoint of a vertical line;
coord2: the y coordinate defines the endpoint of the line. The x coordinate will be ignored.

E.g.: /LINE /VERTICAL 90 110 60 180 130 160 180 120 ;

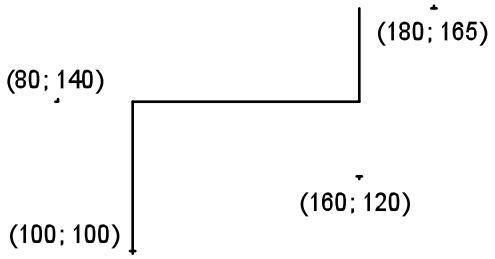


LINE HV < < coord > >

Defines a series of connecting horizontal or vertical lines, depending on the direction defined by the endpoint of the previous line and the latest defined point. If this direction is closer to horizontal, the line will be horizontal, if closer to vertical, it will be vertical. The startpoint of a segment is the first defined point or the endpoint of the previous line, the endpoint is defined by the x- or y-coordinate of the defined point in the case of horizontal or vertical lines respectively.

coord: The first *coord* defines the startpoint of the series of horizontal-vertical lines. The following *coord*-s define the direction and the endpoints of the segments. Enter completes a series of lines and starts a new sequence. Giving Enter again completes the command.

E.g.: /LINE /HV 100 100 80 140 160 120 180 165 ; ;

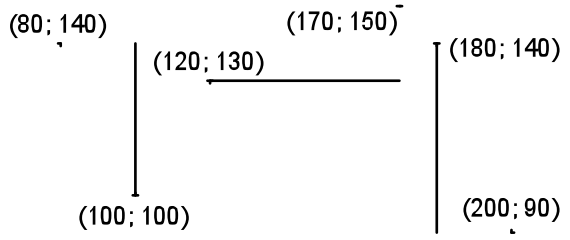


LINE SINGLE HV < coord1 coord2 >

Defines a series of single, horizontal or vertical lines, depending on whether the *coord1 - coord2* direction is closer to horizontal or vertical.

coord1: the startpoint of the line;
coord2: the x or y component of this pair of coordinates defines the endpoint of the horizontal or vertical line respectively.

E.g.: /LINE /SINGLE /HV 100 100 80 140 120 130 170 150 180 140 200 90;



LINE BITANGENT < single1 single2 >

Defines a line tangent to two chosen objects. The endpoints of the line are the tangent points nearest to the selected points.

single1: a point near the first selected object;
single2: a point near the other selected object.

E.g.: Create a circle and an arc

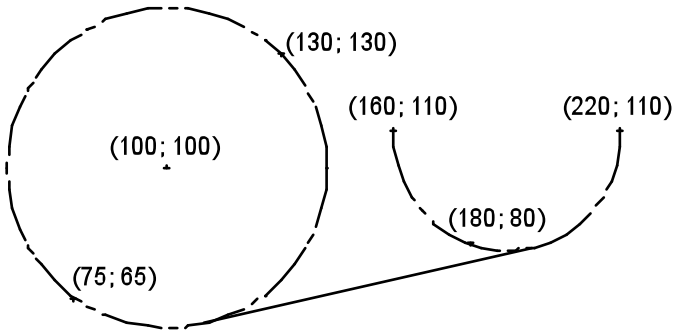
/CIRCLE /CPOINT 100 100 130 130 ;

Create an arc

/CARC /P3 160 110 180 80 220 110 ;

/LINE /BITANGENT 75 65 180 80 ;

Draws a line tangent to the circle and the arc.



LINE PTANGENT < coord < single > >

Draws lines from a specified startpoint tangent to the selected objects. The endpoint of a line is the tangent point. Additional lines can be drawn from the same startpoint if there is more than one possible tangent point.

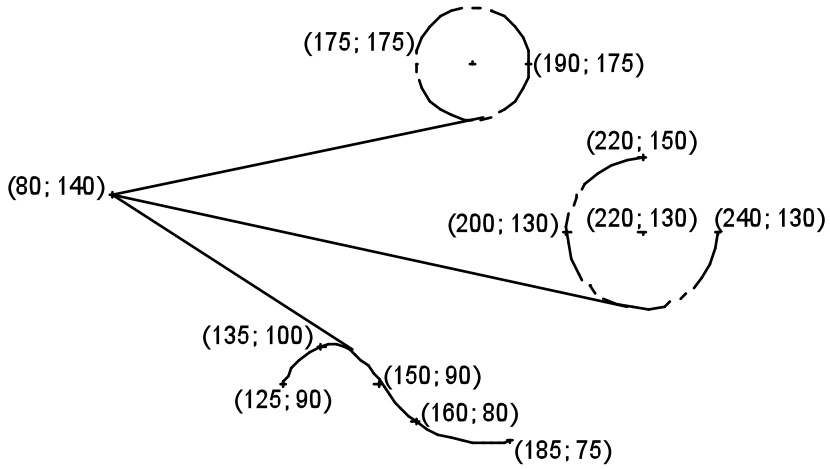
coord: the starting point of the tangent lines;
single: a point near the object the line will be tangential to. Enter completes drawing lines from the current starting point and restarts with another one. Giving ENTER again completes the command.

E.g.: Draw a circle:
/CIRCLE /CPOINT 175 175 190 175 ;

Draw an arc:
/CARC /CPOINT 220 130 220 150 240 130 ;

Draw a spline:
/SPLINE 125 90 135 100 160 80 185 75 ;

Draw tangent lines from the point (80; 140) to all the three objects:
/LINE /PTANGENT 80 140 190 175 200 130 150 90 ; ;

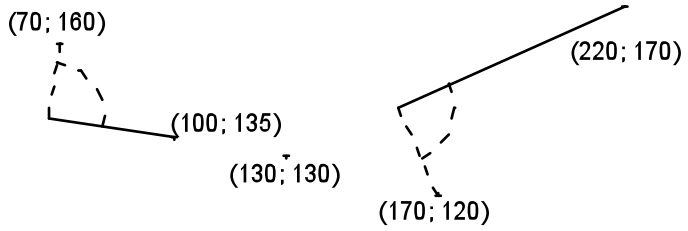


LINE AXIS < coord1 < coord2 coord3 > >

Defines lines passing through the first point, starting from the second point and ending where it comes nearest to the third point.

coord1: The axis point;
coord2: The startpoint of the line. Enter completes drawing lines through the current axis and restarts with another one. Giving ENTER again completes the command.
coord3: The point defining the endpoint of the line.

E.g.: /LINE /AXIS 130 130 100 135 70 160 220 170 170 120 ; ;

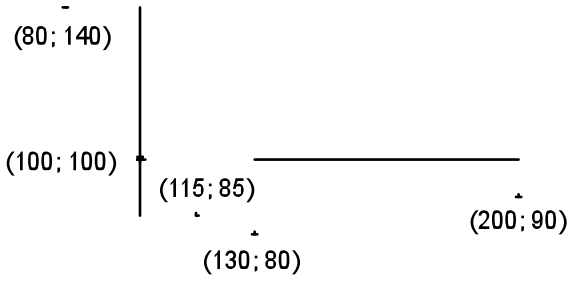


LINE AXIS HV < coord1 < coord2 coord3 > >

Defines horizontal or vertical lines passing through the first point. A line will be horizontal or vertical depending on whether the coord1- coord2 direction is closer to horizontal or vertical. Its endpoints will be the points where it comes nearest to points 2 and 3.

coord1: The axis point.
coord2: Defines the startpoint of a line. Enter completes drawing lines through the current axis point and restarts with another one. Giving ENTER again completes the command.
coord3: Defines the endpoint of a line.

E.g.: /LINE /AXIS /HV 100 100 200 90 130 80 80 140 115 85 ; ;

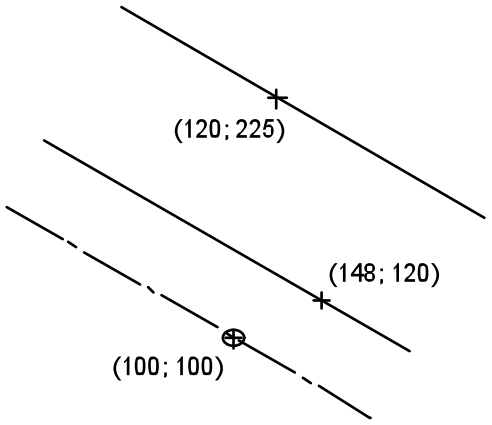


LINE PARALLEL < single < coord > >

Defines lines passing through the second defined point, aligned to an object nearest to the first given point. The length of the parallel lines will be the same as that of the first selected object.

single: a point nearby the object the line will be aligned to;
coord: a point of the parallel line. Enter completes drawing lines aligned to the selected object and restarts with selecting another object. Giving ENTER again completes the command.

E.g.: /LINE /PARALLEL 100 100 148 120 120 225 ; ;



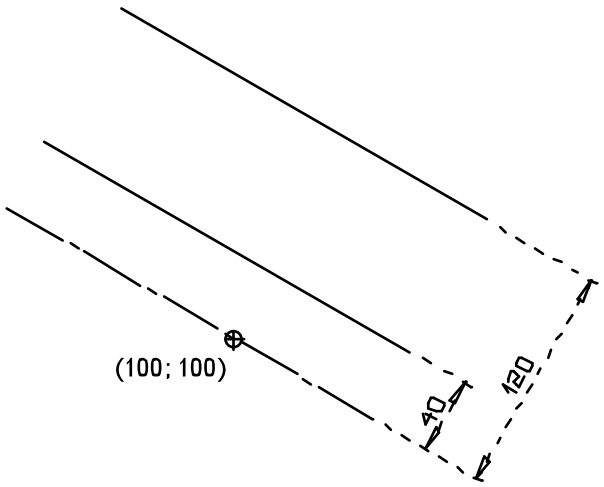
LINE DPARALLEL < single < length > >

Defines lines aligned to an object nearest to the given point, at given distances from the nearest point of the object to the given one.

Positive distance values represent that side of the object where the chosen point is situated. The length of the parallel lines will be the same as that of the first selected object.

single: a point nearby the object the line will be aligned to;
length: the distance of the line from the object. Enter restarts the command with selecting a new object. Giving Enter again completes the command.

E.g.: /LINE /DPARALLEL 100 100 40 120 ; ;

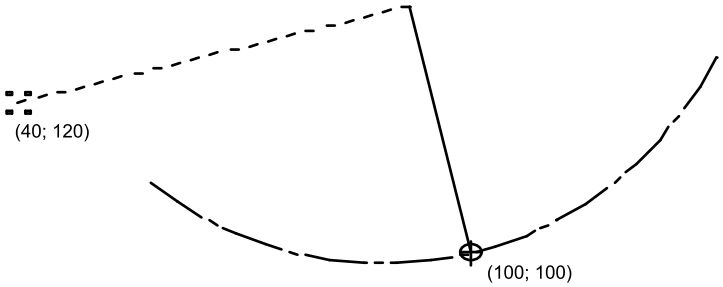


LINE LPERPENDICULAR < single < coord > >

Defines a line starting from the selection point of an object and perpendicular to that object. The endpoint of the line is defined by the perpendicular projection of the pointer.

single: a point on the object the line will be started from;
coord: a point to define the other endpoint.

E.g.: /LINE /LPERPENDICULAR 100 100 40 120 ; ;



BEVEL < length < single1 single2 > >

Draws lines which intersect two non-parallel lines creating a bevel, or chamfer. The startpoint of the bevel is located on the first selected object at a specified distance from the intersection (or apparent intersection) with the second object. The angle between the bevel and the first selected object is the *Preferred Angle*. The command mutilates or extends both objects if necessary.

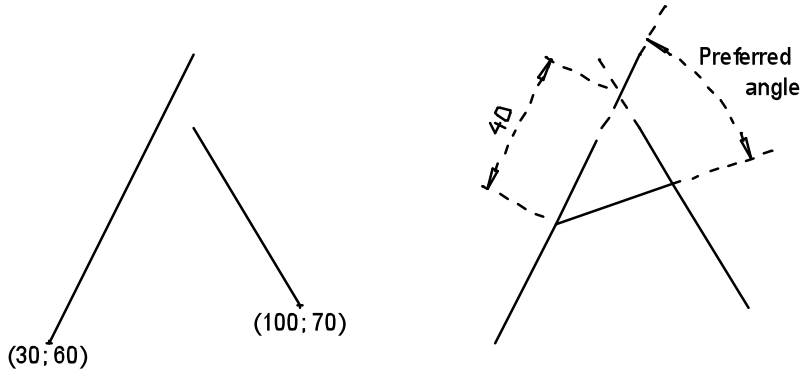
length: The distance of the bevel from the intersection point measured along the line of the first selected object.

single1: A point nearby the object to be selected first. Enter completes the bevel using the given distance and restarts with another one. Giving ENTER again completes the command.

single2: A point nearby the object to be selected second.

E.g.: Create two lines:
/LINE /SINGLE 30 60 70 140 100 70 70 120 ;

Create a bevel between the two lines:
/BEVEL 40 30 60 100 70 ; ;



Before After

Construction Line Defining Commands

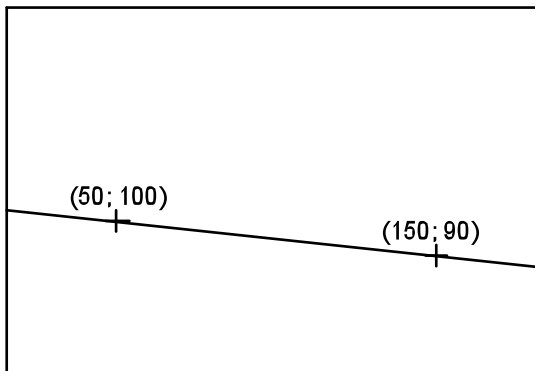
The construction line is a special kind of line which is drawn from one edge of the drawing window to the other.

CONSTLINE < coord1 coord2 >

Defines a construction line by two points.

coord1: a point on the construction line;
coord2: another point on the construction line.

E.g.: /CONSTLINE 50 100 150 90 ;

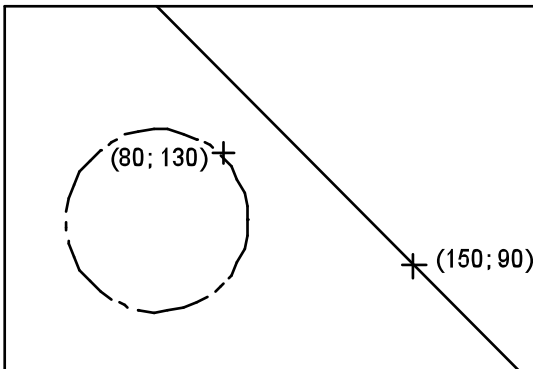


CONSTLINE PARALLEL < single < coord > >

Defines construction lines passing through the second defined point, aligned to an object nearest to the first given point.

single: a point nearby the object the construction line will be aligned to;
coord: a point on the construction line. Enter completes drawing construction lines aligned to the selected object and restarts with selecting another object. Giving ENTER again completes the command.

E.g.: /CONSTLINE /PARALLEL 80 130 150 90 ; ;



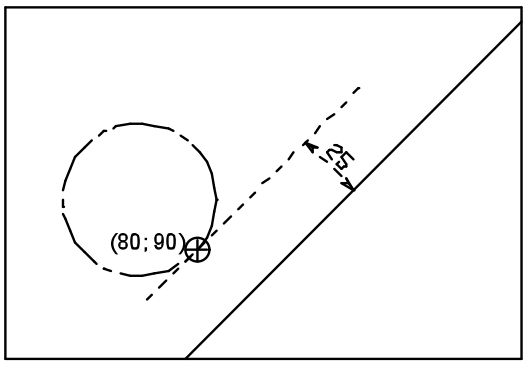
CONSTLINE DPARALLEL < single < length > >

Defines construction lines aligned to an object nearest to the given point, at given distances from the nearest point of the object to the given one.

Positive distance values represent that side of the object where the chosen point is situated.

single: a point nearby the object the construction line will be aligned to;
length: the distance of the construction line from the object. Enter restarts the command with selecting a new object. Giving Enter again completes the command.

E.g.: /CONSTLINE /DPARALLEL 80 90 25 ; ;

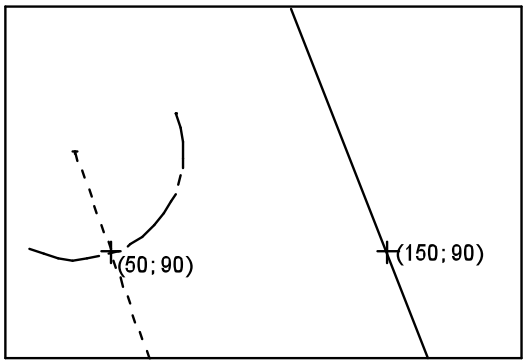


CONSTLINE PERPENDICULAR < single < coord > >

Defines construction lines passing through the given points and perpendicular to the object nearest to the selected point.

single: a point nearby the object the construction line will be perpendicular to;
coord: a point of the construction line. Enter completes drawing lines perpendicular to the given object and starts with selecting another object. Giving Enter again completes the command.

E.g.: /CONSTLINE /PERPENDICULAR 50 90 150 90 ; ;

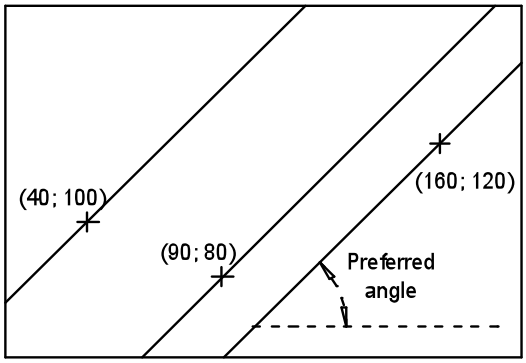


CONSTLINE DFPOINT < coord >

Defines construction lines passing through the given points at the *Preferred Angle*.

coord: a point on the construction line.

E.g.: /CONSTLINE /DFPOINT 40 100 90 80 160 120 ;

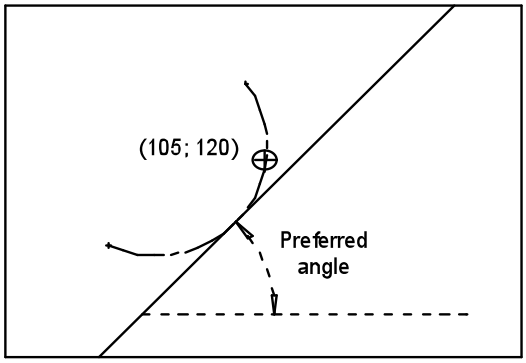


CONSTLINE DFTANGENT < single >

Defines a construction line whose angle is determined by the *Preferred Angle* and tangent to the selected object nearest to the selected point.

single: a point nearby the chosen object.

E.g.: /CONSTLINE /DFTANGENT 105 120 ;

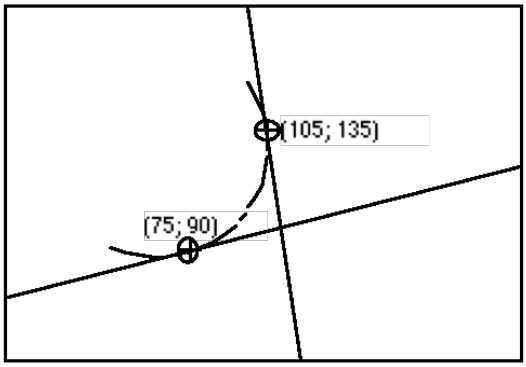


CONSTLINE TANGENT < single >

Defines construction lines tangent to the selected object nearest to the chosen point.

single: a point nearby the chosen object.

E.g.: /CONSTLINE /TANGENT 105 135 75 90 ;

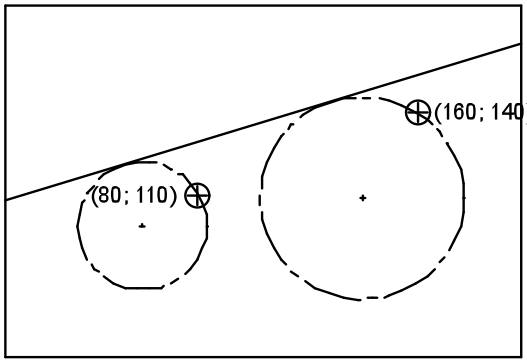


CONSTLINE BITANGENT < single1 single2 >

Defines construction lines tangent to two selected objects nearest to the chosen points.

single1: a point nearby one of the chosen objects;
single2: a point nearby the other chosen object.

E.g.: /CONSTLINE /BITANGENT 80 110 160 140 ;

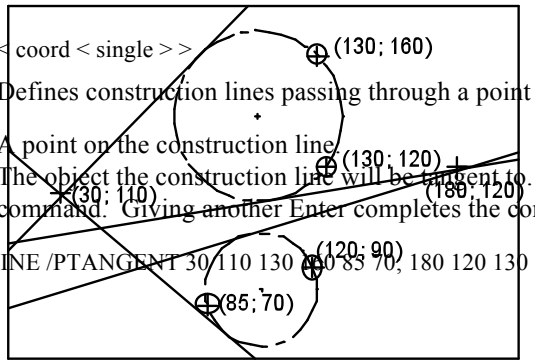


CONSTLINE PTANGENT < coord < single > >

Defines construction lines passing through a point and tangent to objects.

coord: A point on the construction line.
single: The object the construction line will be tangent to. Enter restarts the command. Giving another Enter completes the command.

E.g.: /CONSTLINE /PTANGENT 30 110 130 160 85 70; 180 120 130 120 120 90 ; ;



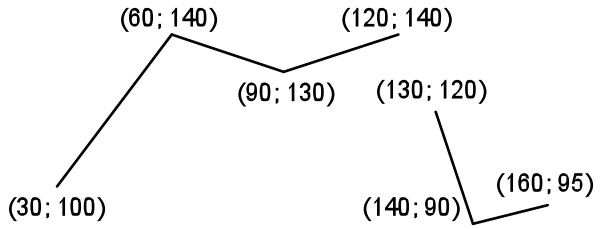
Polyline Defining Commands

POLYLINE [DOUBLE length] < < coord > >

Defines single or double polylines that pass through the specified nodes.

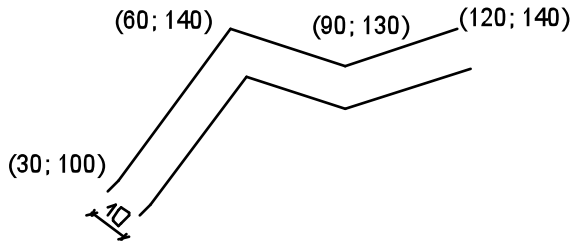
DOUBLE: creates a double polyline;
length: the distance between the two polylines;
coord: a node of a polyline. Enter completes a polyline (to the last defined node) and starts another one. Giving Enter again completes the command.

E.g.: /POLYLINE 30 100 60 140 90 130 120 140 ; 130 120 140 90 160 95 ; ;



E.g.: /POLYLINE /DOUBLE 10 30 100 60 140 90 130 120 140 ; ;

Draws two parallel polylines. The leftside polyline passes through the (30; 100), (60; 140), (90; 130) and (120; 140) points. The distance between the two polylines is 10 units.

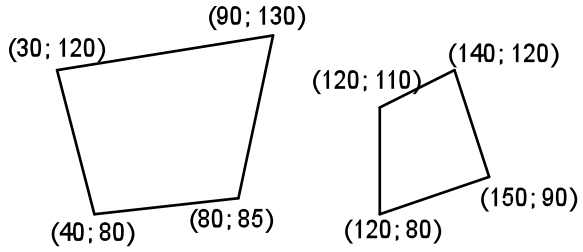


POLYGON [DOUBLE length] < < coord > >

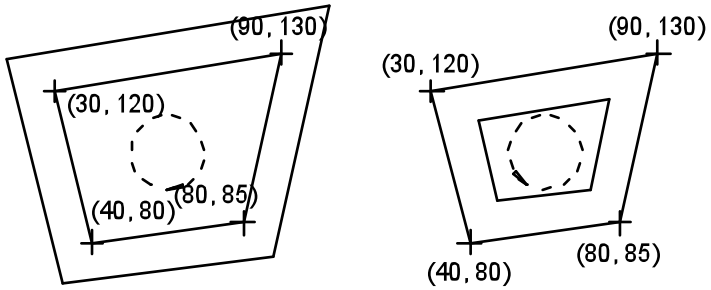
Draws either single or double–line irregular polygons by the vertices. If the DOUBLE keyword is missing it creates single–line polygons, otherwise a double–line polygon is created.

DOUBLE: Creates a double-line polygon;
length: the distance between the internal and external polygons;
coord: a vertex of the polygon. ENTER completes the polygon (connecting the first and last vertices) and starts a new polygon. Giving Enter again completes the command.

E.g.: /POLYGON 30 120 40 80 80 85 90 130 ; 120 110 120 80 150 90 140 120 ; ;



E.g.: the order of points influences the result as is shown in the drawing below
/POLYGON /DOUBLE 10 30 120 40 80 80 85 90 130 ; ; on the left
/POLYGON /DOUBLE 10 30 120 90 130 80 85 40 80 ; ; on the right



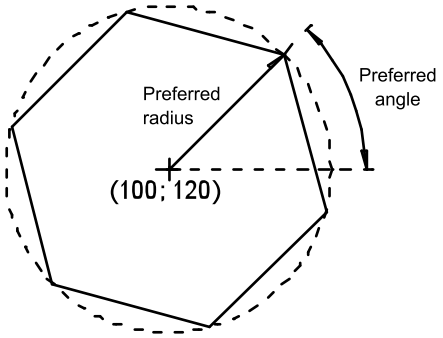
POLYGON REGULAR [{ IN | OUT }] [DOUBLE length] < num < coord > >

Defines a single or double–line regular polygon that is inscribed in or circumscribed around a circle of the *Preferred Radius*. The polygon is rotated by the *Preferred Angle* (a vertex of the polygon will be at the *Preferred Angle* from its center). The number of polygon edges and the location of the center point is specified.

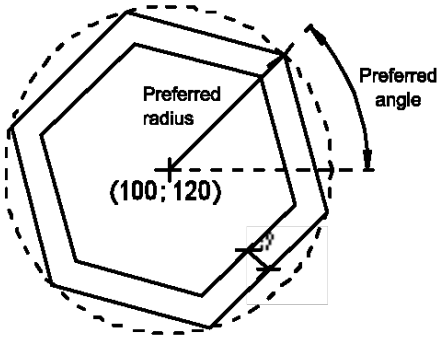
If the keyword IN or OUT is missing, the default is IN.

IN: Defines an inscribed regular polygon;
OUT: defines a circumscribed regular polygon;
DOUBLE: creates a double line polygon;
length: the distance between the inner and outer polygons;
num: the number of edges;
coord: the center of the polygon(s). Enter starts a new polygon with a new value of number of edges. Giving Enter again completes the command.

E.g.: /POLYGON /REGULAR /IN 6 100 120 ; ;



E.g.: /POLYGON /REGULAR /IN /DOUBLE 5 6 100 120 ; ;

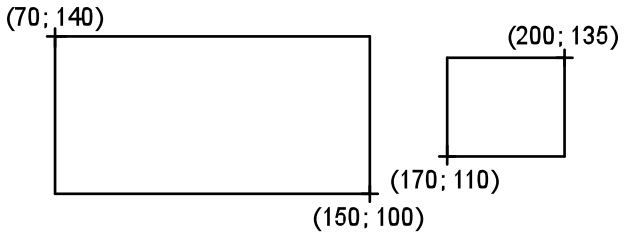


RECTANGLE [DOUBLE length] < coord1 coord2 >

Defines single or double-line rectangles by their opposite corners.

DOUBLE: Creates a double-line rectangle;
length: the distance between the inner and outer rectangles;
coord1: a cornerpoint of the (inner) rectangle;
coord2: the opposite corner of the (inner) rectangle.

E.g.: /RECTANGLE 70 140 150 100 170 110 200 135 ;

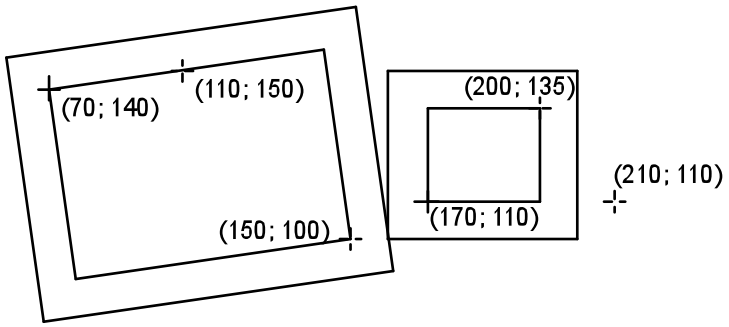


RECTANGLE P3 [DOUBLE length] < coord1 coord2 coord3 >

Defines single or double-line rectangles by three points including their opposite corners.

DOUBLE: Creates a double-line rectangle;
length: the distance between the inner and outer rectangles;
coord1: a cornerpoint of the (inner) rectangle;
coord2: a point to set the orientation of the rectangle;
coord2: the opposite corner of the (inner) rectangle.

E.g.: /RECTANGLE /P3 /DOUBLE 70 140 110 150 150 100 170 110 210 110 200 135 ;

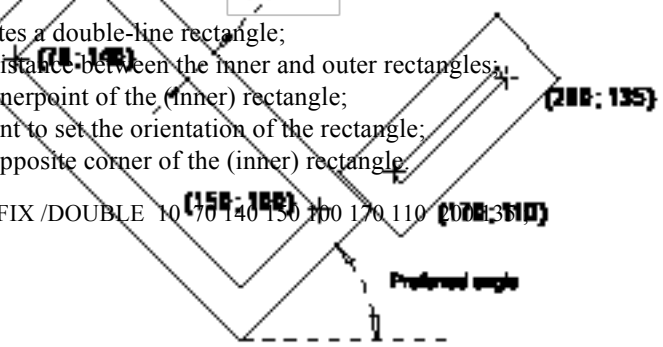


RECTANGLE DFIX [DOUBLE length] < coord1 coord2 >

Defines single or double-l gles by two points. The angle of rectangles is determined b *rred Angle*.

DOUBLE: Creates a double-line rectangle;
length: the distance between the inner and outer rectangles;
coord1: a cornerpoint of the (inner) rectangle;
coord2: a point to set the orientation of the rectangle;
coord2: the opposite corner of the (inner) rectangle.

E.g.: /RECTANGLE /DFIX /DOUBLE 10 70 140 150 100 170 110



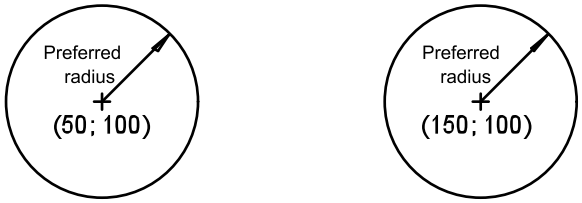
Circle Defining Commands

CIRCLE < coord >

Draws circles at the specified center points whose radius is the *Preferred Radius*.

coord: the center of the circle.

E.g.: /CIRCLE 50 100 150 100 ;

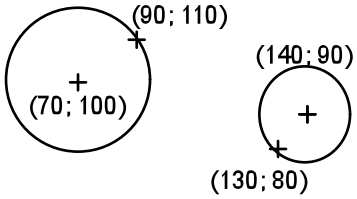


CIRCLE CPOINT < coord1 coord2 >

Defines circles by the center and a point on the perimeter.

coord1: the center point;
coord2: a point on the circle.

E.g.: /CIRCLE /CPOINT 70 100 90 110 140 90 130 80 ;

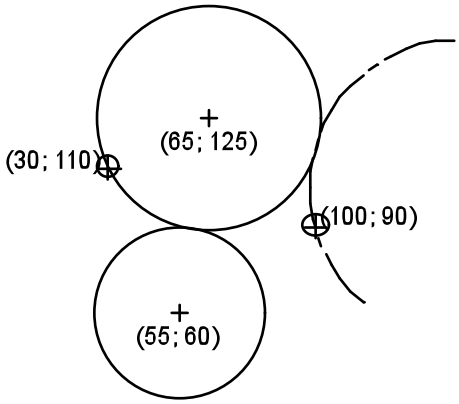


CIRCLE CTANGENT < coord single >

Defines circles with a given center and tangent to the given object (nearest to the chosen point).

coord: the center point;
single: a point near the object the circle will be tangent to.

E.g.: /CIRCLE /CTANGENT 65 125 100 90 55 60 30 110 ;

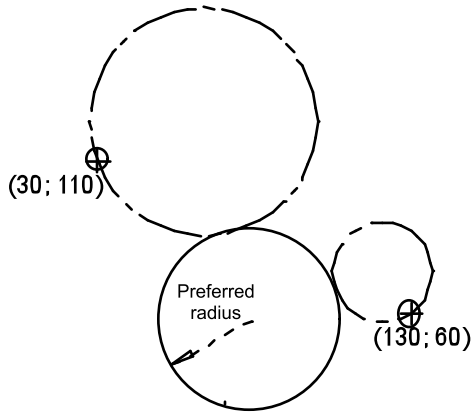


CIRCLE BITANGENT < single1 single2 >

Defines circles with the *Preferred Radius*, tangent to both given objects (nearest to the chosen points).

single1: a point near one of the chosen objects;
single2: a point near the other chosen object.

E.g.: /CIRCLE /BITANGENT 30 110 130 60 ;

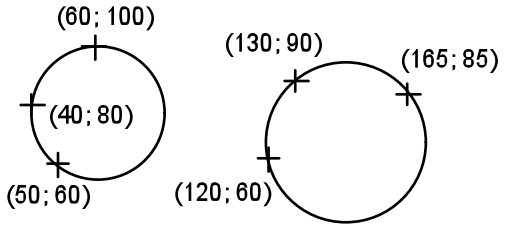


CIRCLE P3 < coord1 coord2 coord3 >

Defines circles by three points.

coord1: a point of the circle;
coord2: another point on the circle;
coord3: the third point on the circle.

E.g.: /CIRCLE /P3 40 80 60 100 50 60 120 60 130 90 165 85 ;

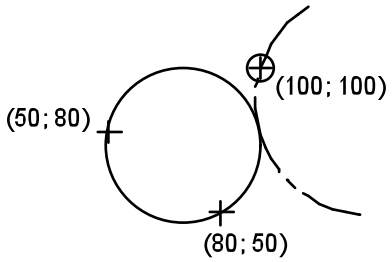


CIRCLE P2TANGENT < coord1 coord2 single >

Defines circles passing through two given points and tangent to the object nearest to the third chosen point.

coord1: a point on the circle;
coord2: another point on the circle;
single: a point near the chosen object.

E.g.: /CIRCLE /P2TANGENT 50 80 80 50 100 100 ;

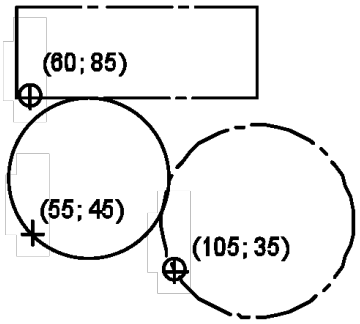


CIRCLE PBITANGENT < coord single1 single2 >

Defines circles passing through a given point and tangent to both given objects (nearest to the chosen points).coord: a point on the circle,

single1: a point near one of the chosen objects;
single2: a point near the other chosen object.

E.g.: /CIRCLE /PBITANGENT 55 45 60 85 105 35 ;

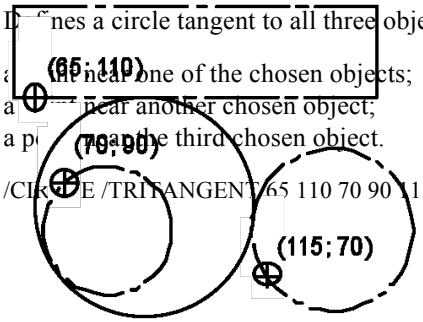


CIRCLE TRITANGENT < single1 single2 single3 >

Defines a circle tangent to all three objects nearest to the chosen points.

single1: a point near one of the chosen objects;
single2: a point near another chosen object;
single3: a point near the third chosen object.

E.g.: /CIRCLE /TRITANGENT 65 110 70 90 115 70 ;



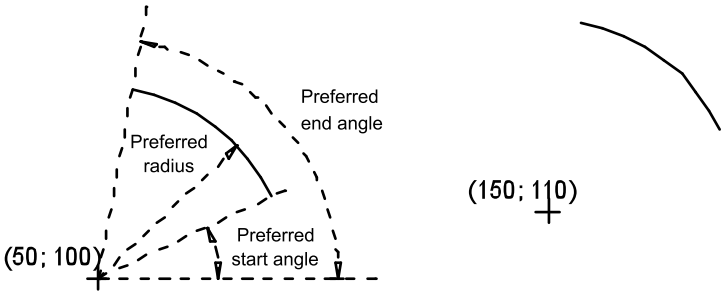
Circular Arc Defining Commands

CARC < coord >

Defines a circular arc at a specified center point. Its radius is the *Preferred Radius*, its endpoints are determined by the *Preferred End Angles*.

coord: the center point of the arc.

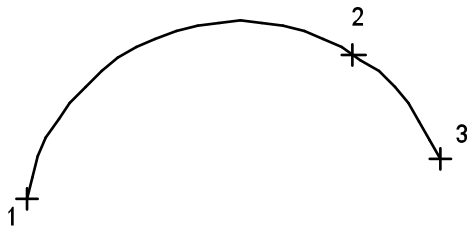
E.g.: /CARC 50 100 150 110 ;



CARC P3 < coord1 coord2 coord3 >

Defines a circular arc that passes through three specified points. The first and third points are the endpoints of the arc.

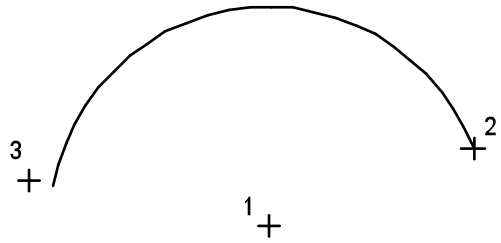
coord1: one of the endpoints of the arc;
coord2: an internal point of the arc;
coord3: the other endpoint of the arc.



CARC CPOINT < coord1 coord2 coord3 >

Defines a circular arc by its center point and the endpoints.

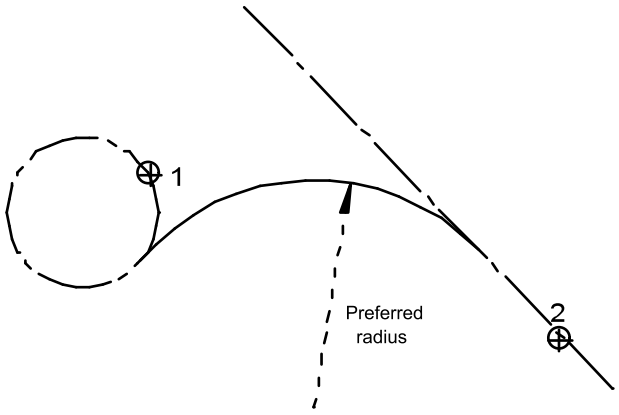
coord1: the center point of the arc;
coord2: an endpoint of the arc;
coord3: the other endpoint of the arc.



CARC BITANGENT < single1 single2 >

Defines a round-off of the *Preferred Radius* between two objects nearest to the chosen points.

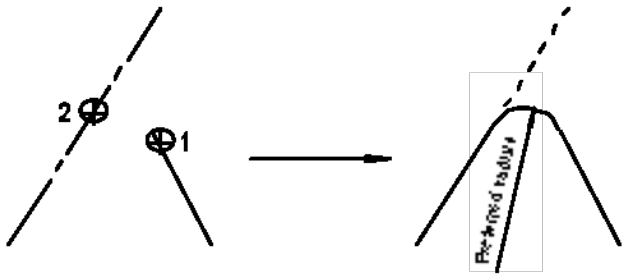
single1: a point near one of the objects;
single2: a point near the other object.



FILLET < single1 single2 >

Defines a round-off, using the *Preferred Radius*, between two selected objects. This command either truncates or adds to the length of the selected objects. The endpoints opposite the round-off remain unchanged.

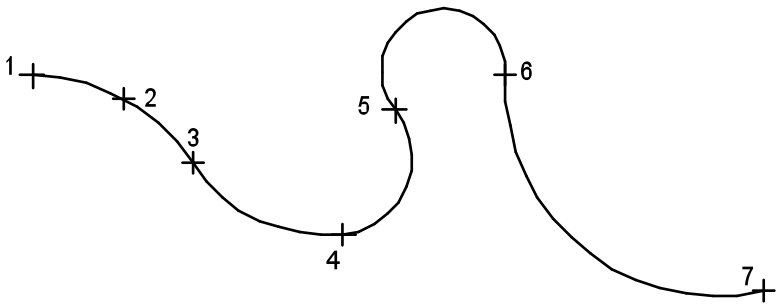
single1: a point near one of the objects;
single2: a point near the other object.



CCURVE [SMOOTH] < coord1 coord2 coord3 < coord4 > >

Defines a smooth curve composed of circular arcs. The first three points define the first circular arc, and the remaining points are the endpoints of the connecting arcs; the connection is smooth.

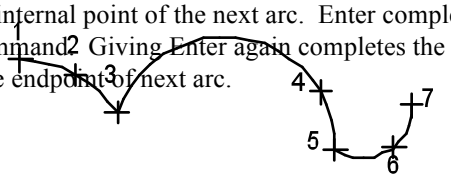
coord1: the starting point of the first arc;
coord2: an internal point of the first arc;
coord3: the endpoint of the first arc;
coord4: the endpoints of the consequent arcs. Enter completes the curve and repeats the command. Giving Enter again completes the command.



CCURVE CONTIGUOUS < coord1 coord2 coord3 < coord4 coord5 > >

Defines a contiguous curve composed of circular arcs. The first three points define the first circular arc, the endpoint of the previous arc and two additional points define the next arc.

coord1: The starting point of the first arc;
coord2: an internal point of the first arc;
coord3: the endpoint of the first arc;
coord4: an internal point of the next arc. Enter completes the curve and repeats the command. Giving Enter again completes the command.
coord5: The endpoint of next arc.

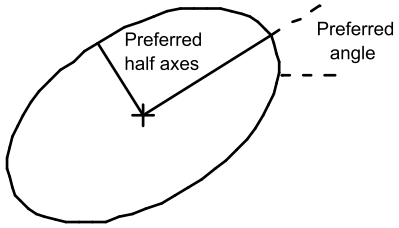


Ellipse Defining Commands

ELLIPSE < coord >

Defines an ellipse at a specified center point. Its size is determined by the *Preferred Half Axes*. The angle of the major axis is the *Preferred Angle*.

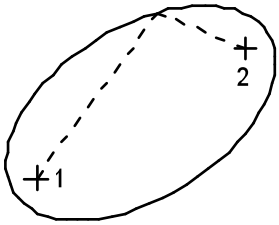
coord: the center point of the ellipse.



ELLIPSE FDISTANCE < coord1 coord2 length >

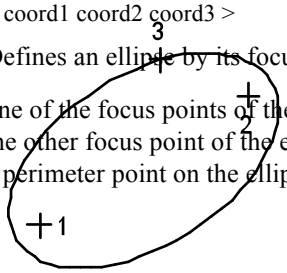
Defines an ellipse by its focus points and the sum of the distances between each focus point and a point on the perimeter.

coord1: one of the focus points of the ellipse;
coord2: the other focus point of the ellipse;
length: length of the major axis of the ellipse.



ELLIPSE FPOINT < coord1 coord2 coord3 >

Defines an ellipse by its focus points and a point on its perimeter.



coord1: one of the focus points of the ellipse;
coord2: the other focus point of the ellipse;
coord3: a perimeter point on the ellipse.

Elliptic Arc Defining Commands

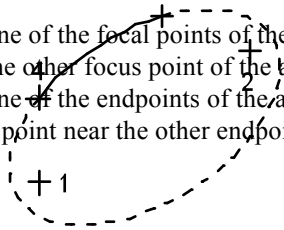
EARC < coord >

Defines an elliptic arc at a specified center point. Its size is determined by the length of the *Preferred Half Axes*, its endpoints are determined by the *Preferred End Angles*. The angle of the major axis is the *Preferred Angle*.

coord: the center point of the elliptic arc.

EARC FPOINT < coord1 coord2 coord3 coord4 >

Defines an elliptic arc specified by its focal points, one of its endpoints, and a nearby point to the other endpoint. The arc is drawn counter-clockwise.



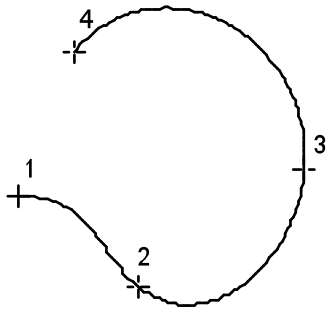
coord1: one of the focal points of the elliptic arc;
coord2: the other focus point of the arc;
coord3: one of the endpoints of the arc;
coord4: a point near the other endpoint of the arc.

Spline Defining Commands

SPLINE [OPENED] < < coord > >

Defines open splines that pass through specified nodes. The length of the end tangents is zero.

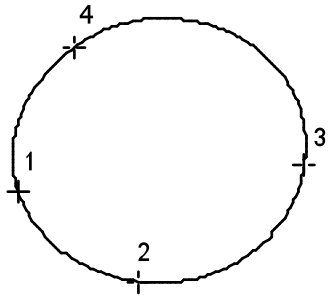
coord: A node of the spline. Enter completes a spline and repeats the command. Giving Enter again completes the command.



SPLINE CYCLIC < < coord > >

Defines closed splines that pass through specified nodes. The tangent vectors in the first and last points are identical.

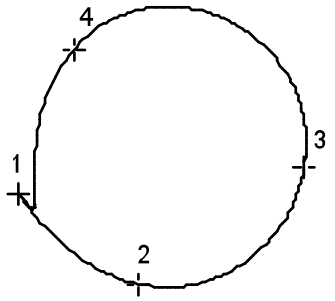
coord: A node of the spline. Enter completes a spline and repeats the command. Giving Enter again completes the command.



SPLINE ANTICYCLIC < < coord > >

Defines a closed spline that passes through specified nodes. The tangent vectors in the first and last points are opposite.

coord: A node of the spline. Enter completes a spline and repeats the command. Giving Enter again completes the command.



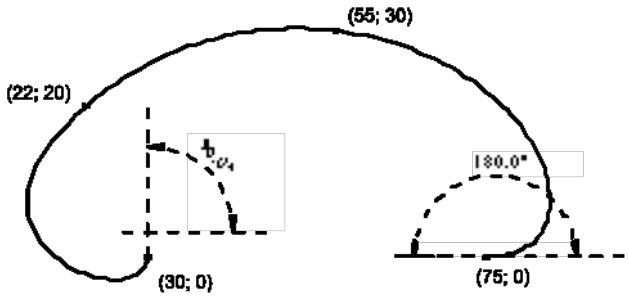
SPLINE TANGENT < length1 angle1 length2 angle2 < coord > >

Defines splines by the tangents at the endpoints and the nodes.

length1: Length of the first tangent vector.
angle1: Angle of the first tangent vector.
length2: Length of the last tangent vector.
angle2: Angle of the last tangent vector.
coord: A node of the spline. Enter completes a spline and repeats the command. Giving Enter again completes the command.

E.g.: /SPLINE /TANGENT 25 90 50 180 30 0 22 20 55 30 75 0 ; ;

Draws a spline with a tangent of 90°, length 100, at the startpoint, and tangent of 180°, length 200, at the endpoint.



Text Defining Command

TEXT < coord >

The command locates the *Preferred Text* at the given positions.

coord: position of the text origin.

E.g.: /TEXT 30 100 50 130 ;

(50; 130)

(30; 100)

E.g.: /TEXT /ATEXT "Sample Text" 92 190 ;

locates “Sample Text” at (92; 190) by redefining the *Preferred Text* temporarily.

E.g.: /TEXT /TDIRECTION 90 145 90 ;

locates the *Preferred Text* vertically at the position (145; 90).

Hatch Defining Commands

Hatch defining commands in topCAD include hatching with lines, patterns, symbols and the so-called composite hatching. Accordingly, the general syntax of the wide variety of hatch defining commands can be divided in four main groups, though most of the syntactical elements are common. Because of this, the syntax of the four groups is given together.

HHATCH { HATCHLINE | HATCHPATTERN | HATCHSYMBOL | HATCHCOMP }
[CHAIN | INTCHAIN | SHAPE | INTSHAPE] { IN | OUT }
[num | name] < group1 [HOLE group2] >

Hatches the area inside or outside closed chains of objects. The method of hatching depends on the keyword following HHATCH. If none of the optional CHAIN, INTCHAIN, SHAPE or INTSHAPE keywords are given, the objects of the chain must be specified one by one. These objects are existing objects in the drawing. If either the SHAPE or the INTSHAPE keyword is given, the hatching area is made of virtual polygons rather than parts of existing objects. num or name must be given when hatching is other than with hatchline.

HATCHLINE: hatches the area using parallel straight lines;

HATCHPATTERN: hatches the area using the specified hatch pattern, num, which identifies the pattern by its serial number and must be included in the command line;

HATCHSYMBOL: hatches the area using the specified hatch symbol, name, which identifies the symbol by name and must be included in the command line;

HATCHCOMP: hatches the area using the specified composite hatch, num, which identifies the composite hatch by its serial number and must be included in the command line;

CHAIN: specifying any object in the chain, the chain is to be recognized by topCAD;

INTCHAIN: specifying any object in the chain, the chain and all internal chains (only one level deep) inside are to be recognized by topCAD;

SHAPE: the outline of the hatching area is specified by a closed chain of straight lines (a polygon). The outline of the hatch area is not drawn.

INTSHAPE: the hatching area and internal areas to be excluded from hatching are specified by a polygon. Neither outlines of the hatching area nor the outline of the excluded area are drawn.

IN: Hatching inside the selected area;

OUT: hatching outside the selected area. The hatching extends to the borderlines of the active drawing window.

num: Serial number of the hatch pattern or the composite hatch;

name: name of the symbol used for hatching;

group1: selection of the area to be hatched, either by selecting objects in the chain or defining virtual polygons (if the SHAPE or INTSHAPE keyword is given). Enter completes the selection of a hatch boundary. Giving Enter again completes the command.

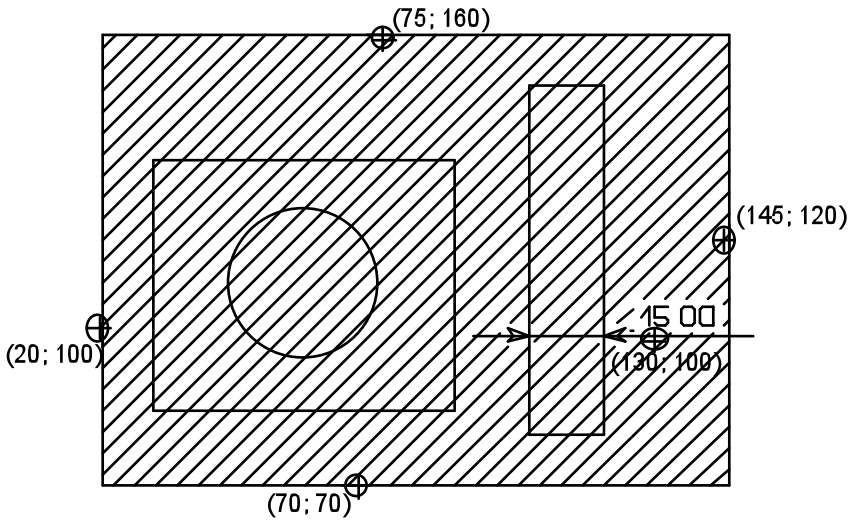
HOLE: This optional keyword must be followed by a list of dimensions and textual objects. A rectangle around these text and dimension type objects will be excluded from hatching.

group2: Selection oftext and dimensions to be excluded from hatching. Enter completes the selection of text and dimensions to be excluded. Giving Enter again completes the command.

Let us see some examples of hatching. The examples do not exhaust the variations of the hatching commands available, but give an overview of the possibilities. Feel free to use any combination of the command alternatives.

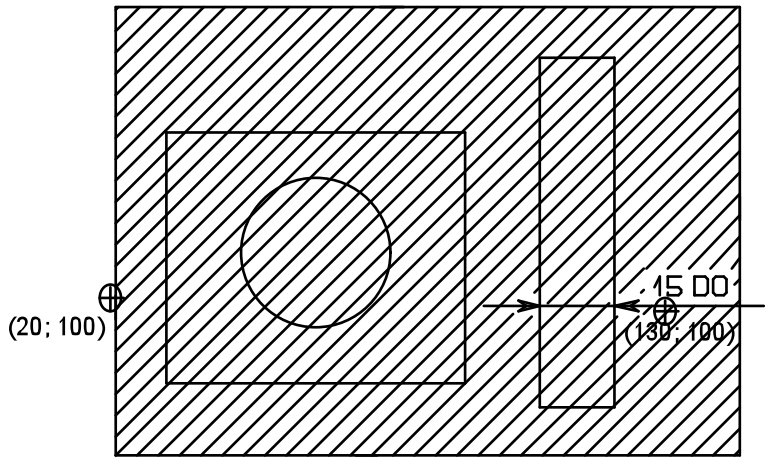
E.g.: Let us hatch an area by defining its border manually selecting the objects in the chain. Hatch the area *inside* the chain with lines excluding the dimension text in the drawing.

/HHATCH /HATCHLINE /IN 20 100 70 70 145 120 75 160 ; /HOLE 130 100 ; ;



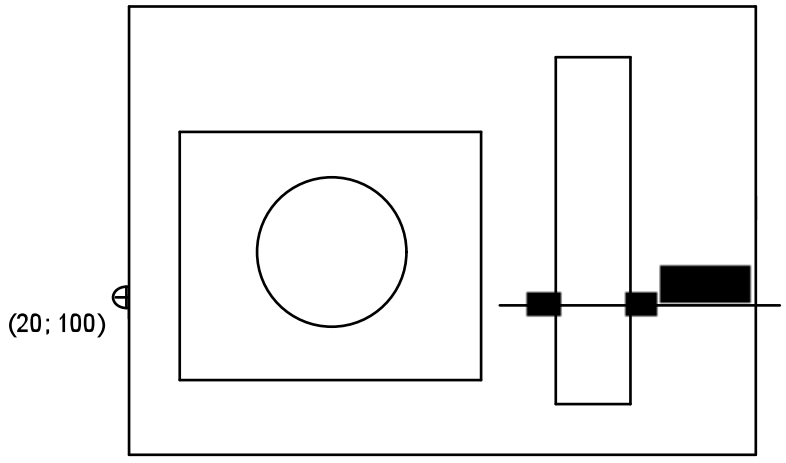
E.g.: In this example the task is the same as in the previous example but here we will have the chain recognized by topCAD:

/HHATCH /HATCHLINE /CHAIN /IN 20 100 /HOLE 130 100 ; ;



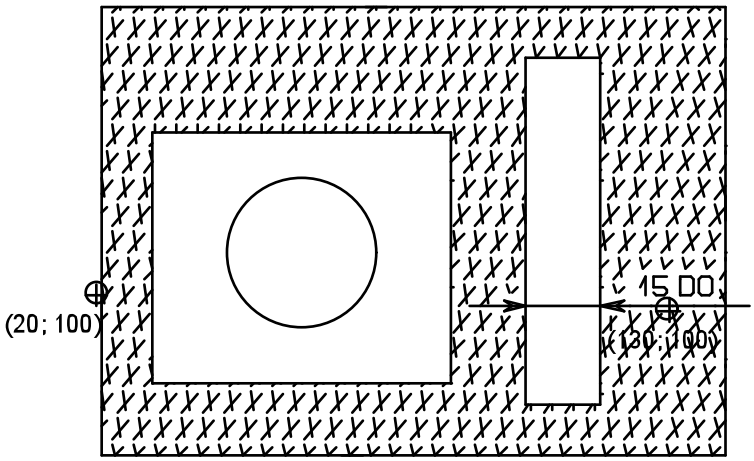
E.g.: Let us hatch with pattern in this example. Use the pattern Nº 61. As for the other details, the task is the same as before. As we do it with hatch pattern, we could do it using symbol or composite hatch.

/HHATCH /HATCHPATTERN /CHAIN /IN 61 20 100 /HOLE 130 100 ; ;



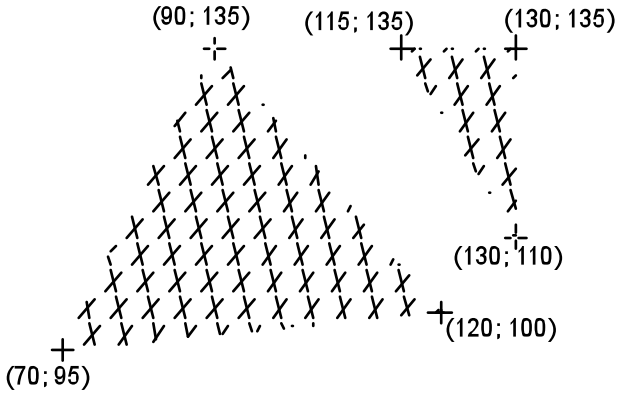
E.g.: In this example let us use a symbol named hatch1 for hatching. Now exclude the first level internal chains from hatching.

/HHATCH /HATCHSYMBOL /INTCHAIN /IN hatch1 20 100 /HOLE 130 100 ; ;



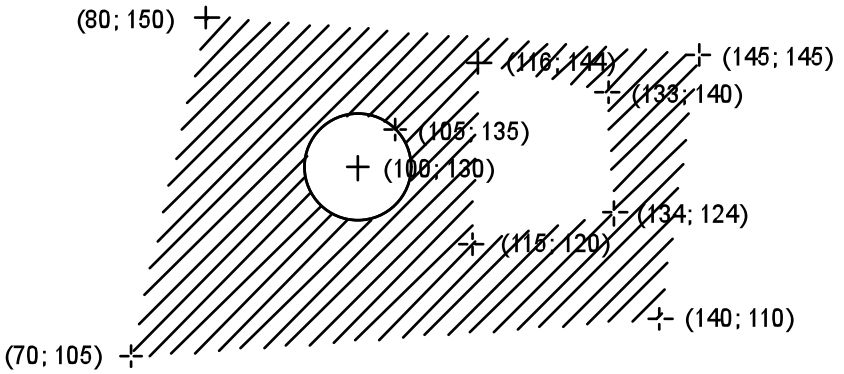
E.g.: So far we have used existing objects to make part of the chain when hatching. Now let us define the hatching boundaries similarly to polylines. Hatch the area *inside* the boundaries. Use the symbol named hatch1 for hatching. (As with the previous examples, any hatching method could be used.)

/HHATCH /HATCHSYMBOL /SHAPE /IN hatch1 70 95 120 100 90 135 ; 115 135 130 135 130 110 ; ;



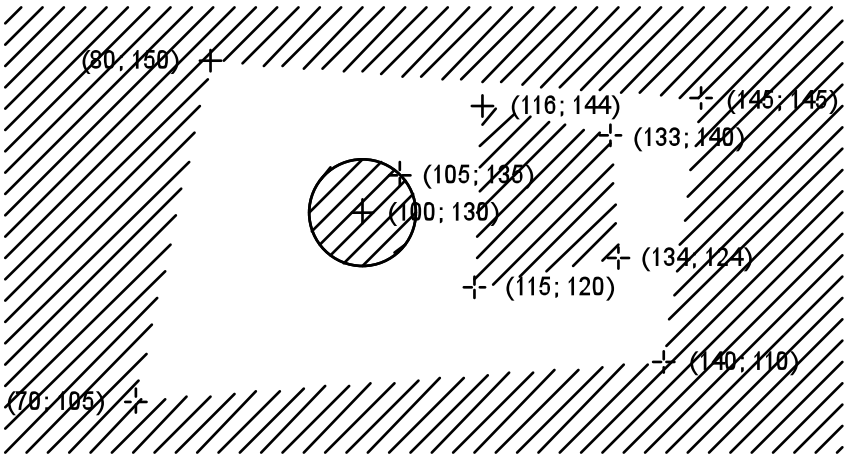
E.g.: Let us hatch inside arbitrary boundaries and have one level of internal closed chains or objects and arbitrary areas automatically excluded. Although the circle in this example is an existing object and the internal rectangle does not exist., both of them are excluded from hatching.

/CIRCLE /CPOINT 100 130 105 135 ;
/HHATCH /HATCHLINE /INTSHAPE /IN 80 150 145 145 140 110 70 105 ; 116 144 133 140 134 124 115 120 ; ;



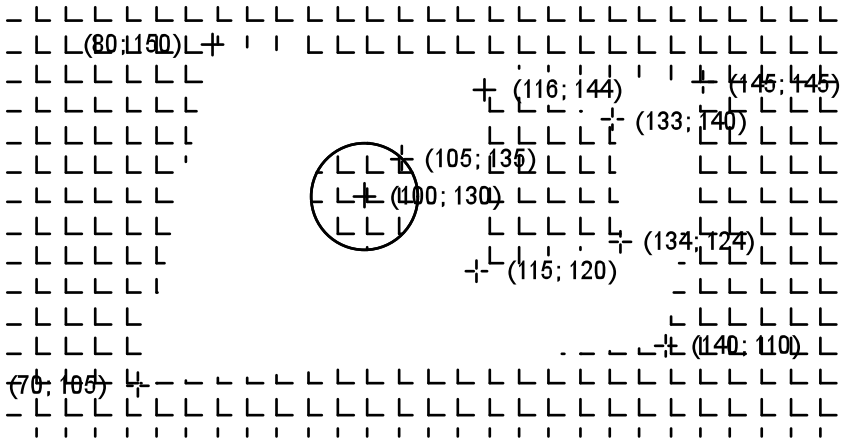
E.g.: Now , the task will be the same as in the previous example but hatch the area outside the boundaries. The example uses the /OUT keyword. Notice that hatching extends to the boundaries of the current window.

/HHATCH /HATCHLINE /INTSHAPE /OUT 80 150 145 145 140 110 70 105 ; 116 144 133 140 134 124 115 120 ; ;



E.g.: In this example we use the previous command but now we hatch with composite lines. Notice, that this kind of hatching requires another parameter, the composite hatch identifier, that is “Angle”.

/HHATCH /HATCHCOMP /INTSHAPE /OUT "Angle" 80 150 145 145 140 110 70 105 ; 116 144 133 140 134 124 115 120 ; ;



E.g.: The same example, but using pattern hatch. An identifier, the pattern serial number, must be given in this case.

/HHATCH /HATCHPATTERN /INTSHAPE /OUT 24 80 150 145 145 140 110 70 105 ; 116 144 133 140 134 124 115 120 ; ;

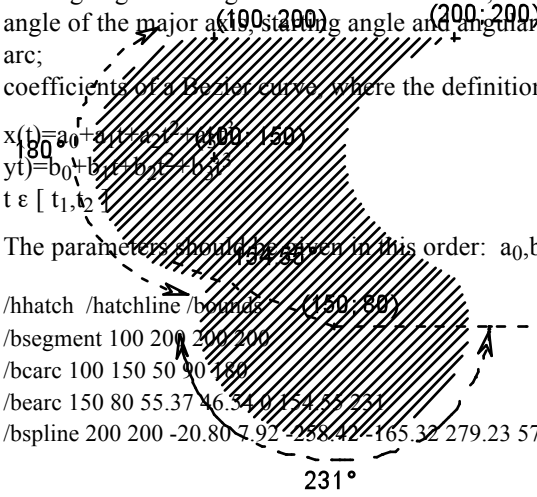
HHATCH [HATCHLINE | HATCHPATTERN | HATCHSYMBOL| HATCHCOMP] BOUNDS < [
BSEGMENT coord1 coord2 |
BCARC coord length angle1 angle2 |
BEARC coord length1 length2 angle1 angle2 angle3 |
BSPLINE num1 num2... num10] >

Hatches the area determined by segments, arcs, elliptic arcs, or splines. In contrast to the other hatch commands this one determines the area to be hatched by objects described in the command itself, rather than with existing objects in the drawing.

coord1, coord2: endpoints of a segment;
coord: centerpoint of a circle or an ellipse;
length, lenght1, length2: radius, and size of half axes;
angle1, angle2: starting angle and angular subtence of an arc;
angle1, angle2, angle3: angle of the major axis, starting angle and angular subtence of an elliptic arc;
num1, num2, ... num10: coefficients of a Bezier curve, where the definition of the curve is:

$x(t)=a_0+a_1t+a_2t^2+a_3t^3$
$yt)=b_0+b_1t+b_2t^2+b_3t^3$
$t \varepsilon [t_1,t_2]$

The parameters should be given in this order: $a_0,b_0,...a,b_3$



E.g.: /hhatch /hatchline /bounds < >
/bsegment 100 200 200 200
/bcarc 100 150 50 90 180
/bearc 150 80 55.37 46.54 0.154 55 231
/bspline 200 200 -20.80 7.92 -258.42 -165.32 279.23 57.40 0.048 1 ; ;

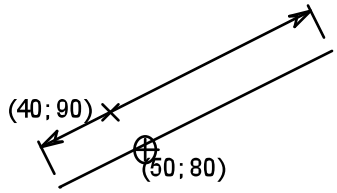
Dimensioning Commands

DIMENSION [{ L | LX | LY }] < single coord >

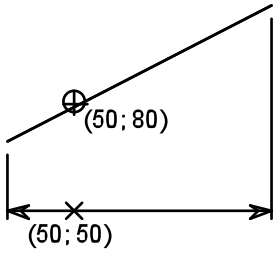
Dimensions the selected objects. The position of the dimension line is defined by a given point.

L: dimensions parallel to the object (this is the default keyword);
LX: dimensions the horizontal projection of the object;
LY: dimensions the vertical projection of the object.
single: the object to be dimensioned;
coord: the position of the dimension.

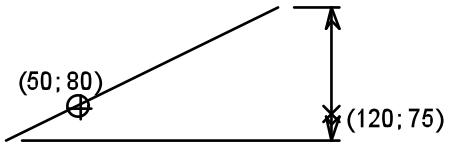
E.g.: /DIMENSION /L 50 80 40 90 ;



E.g.: /DIMENSION /LX 50 80 50 50 ;



E.g.: /DIMENSION /LY 50 80 120 75 ;

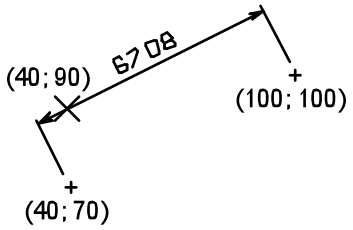


DIMENSION { DISTANCE | DISX | DISY } < coord1 coord2 coord3 >

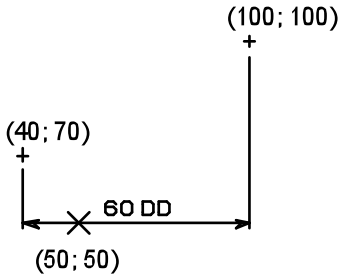
Dimensions the distance between two selected points or its projections.

DISTANCE: dimensions the distance;
DISX: dimensions the horizontal projection of the distance;
DISY: dimensions the vertical projection of the distance;
coord1: a point near the first chosen point;
coord2: a point near the other chosen point;
coord3: position of the dimension.

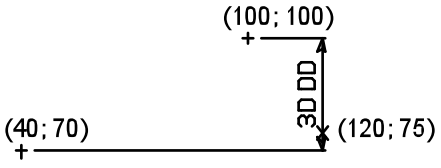
E.g.: /DIMENSION /DISTANCE 40 70 100 100 40 90 ;



E.g.: /DIMENSION /DISX 40 70 100 100 50 50 ;



E.g.: /DIMENSION /DISY 40 70 100 100 120 75 ;

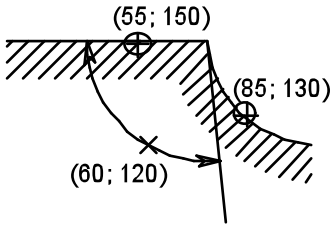


DIMENSION { DANGLE | CANGLE } < single1 single2 coord >

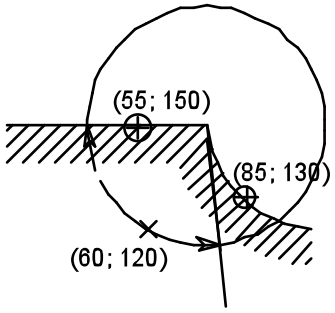
Dimensions the angle between two objects at the intersection point, regardless of which one is selected first.

DANGLE: dimensions the smaller angle between the objects;
CANGLE: dimensions the complementary angle between the objects;
single1: a point near one of the objects;
single2: a point near the other object;
coord: position of the dimension.

E.g.: /DIMENSION /DANGLE 55 150 85 130 60 120 ;



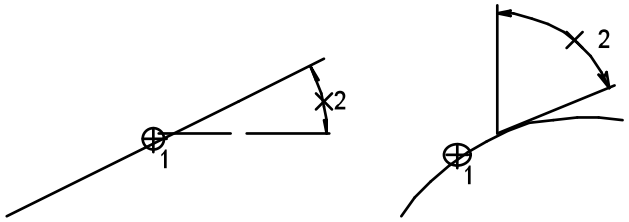
E.g.: /DIMENSION /CANGLE 55 150 85 130 60 120 ;



DIMENSION { AX | AY | CAX | CAY } < single coord >

Dimensions the angle between the object and the horizontal or vertical axis (determined by the position of the pick) or the complementary angle. The dimension line passes through the given point.

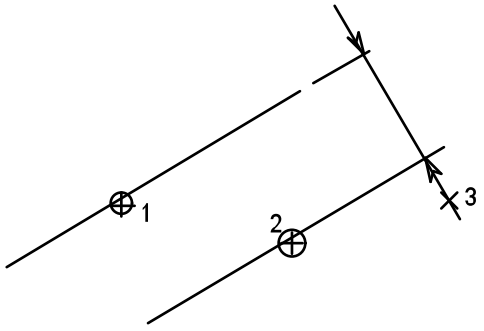
AX: dimensions the angle between the object and the +x-axis;
AY: dimensions the angle between the object and the +y-axis;
CAX: dimensions the complementary angle between the object and the +x-axis;
CAY: dimensions the complementary angle between the object and the +y-axis;
single: a point near the object to be dimensioned;
coord: a point locating the dimension.



DIMENSION PARALLEL < single1 single2 coord >

Dimensions the distance between two selected parallel lines.

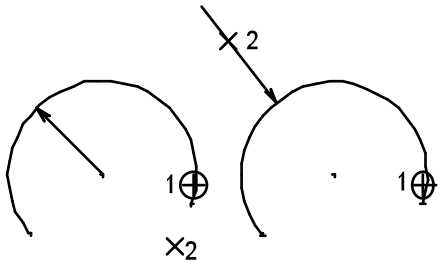
single1: a point near one of the objects to be dimensioned;
single2: a point near the other object;
coord: a point locating the dimension.



DIMENSION DRADIUS < single coord >

Dimensions the radius outside or inside a circle or circular arc. The inside dimension line starts from the center point of the circle or circular arc and the text is in the bisector; dimensioned outside, the text is located above the given point.

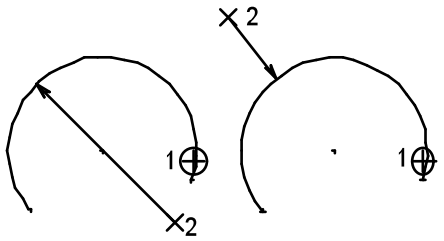
single: a point near the object to be dimensioned;
coord: a point locating the dimension.



DIMENSION RINTERNAL < single coord >

Dimensions the radius outside or inside a circle or circular arc. The text is above the given point.

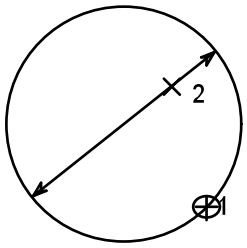
single: a point near the object to be dimensioned;
coord: a point locating the dimension.



DIMENSION DIAMETER < single coord >

Dimensions the diameter outside or inside a circle. The text is above the given point. The dimension line passes through the given point.

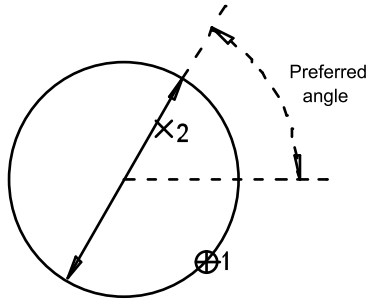
single: a point near the object to be dimensioned;
coord: a point locating the dimension.



DIMENSION DFIX < single coord >

Dimensions the diameter outside or inside a circle. The direction of the dimension line is the *Preferred Direction*. The distance of the given point from the center point of the circle indicates the position of the dimensioning text.

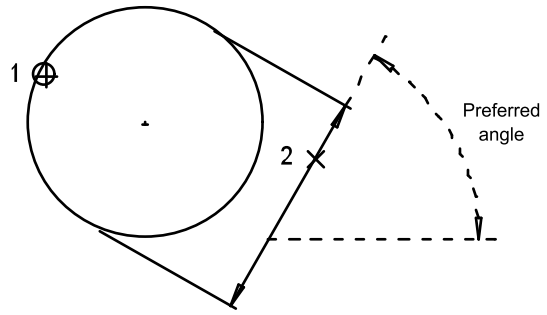
single: a point near the object to be dimensioned;
coord: a point locating the dimension.



DIMENSION DEXTERNAL < single coord >

Dimensions the diameter outside a circle. The direction of the dimension line is the *Preferred Direction*. The dimension line passes through the given point.

single: a point near the object to be dimensioned;
coord: a point locating the dimension.

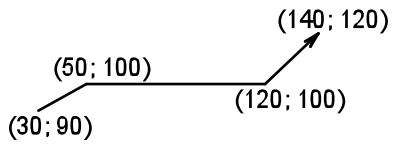


DIMENSION ARROW < coord1 coord2 { ENTER | coord3 { ENTER | coord4 } } >

Defines a leader line without any text. The number of the segments may vary from 1 to 3. The arrow is at the end of the line.

coord1: The starting point of the first segment;
coord2: the endpoint of the first segment.
coord3: The endpoint of the second segment. Enter completes a one-segment leader line, and restarts the command. Giving Enter again completes the command.
coord4: The endpoint of the three-segment leader line. The command restarted. Enter completes a two-segment leader line and restarts the command. Giving Enter again completes the command.

E.g.: **/DIMENSION ARROW 30 90 50 100 120 100 140 120 ;**

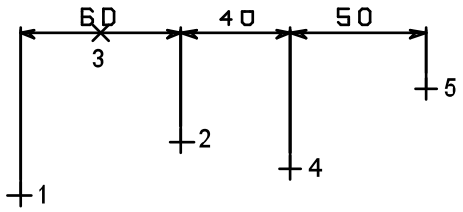


DIMENSION { DISTANCE | DISX | DISY }
{ SERIAL | CUMULATIVE | PARALLEL | PROGRESSIVE | SPART } < coord1 coord2 coord3 < coord4 > >

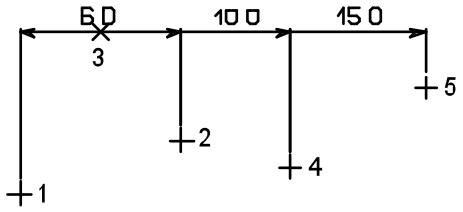
Defines a serial (continuous), parallel, cumulative (baseline), progressive (ordinate) or half-diameter dimension group. The interpretation of the first three points is identical to the case of normal dimensioning between two points. Entering additional points, additional dimensions are created (in the case of parallel dimensioning the distance between the two neighboring dimension lines depends on the character height).

DISTANCE: defines dimensions parallel to an imaginary line passing through the first two points selected ;
DISX: defines horizontal dimensions;
DISY: defines vertical dimensions;
SERIAL: defines continuous dimensions;
CUMULATIVE: defines baseline dimensions;
PARALLEL: defines parallel dimensions;
PROGRESSIVE: defines ordinate dimensions;
SPART: defines half–diameter dimensions;
coord1: starting point of the dimensioning;
coord2: a point near the first point to be dimensioned from coord1;
coord3: a point locating the dimension;
coord4: points near the next points to be dimensioned. Enter completes a series of dimensions and restarts with a new starting point.

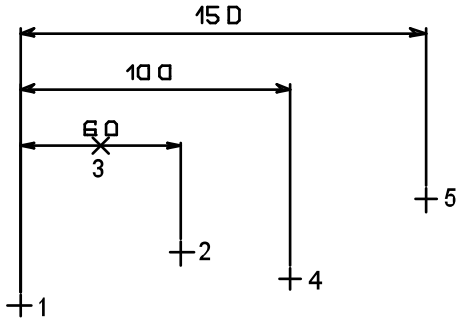
SERIAL **(continuous, horizontal):**



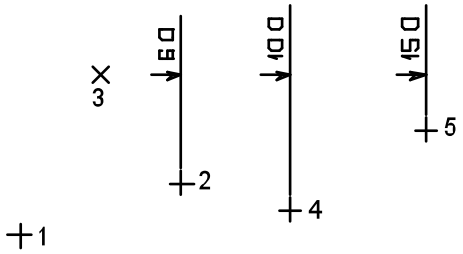
CUMULATIVE **(baseline, horizontal):**



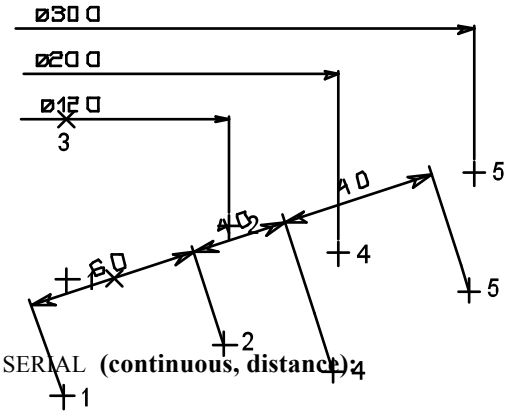
PARALLEL **(parallel, horizontal):**



PROGRESSIVE **(ordinate, horizontal):**



SPART **(half-diameter, horizontal):** for dimensioning cylindric holes or objects



SERIAL **(continuous, distance):**

Symbol Defining Commands

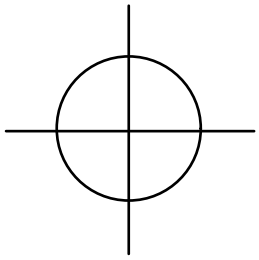
SYMBDEF name group coord

Creates a symbol with the given name. The symbol comprises the selected group of objects, its origin is the given point.

Replaces the unresolved symbol references of the given name with the newly created symbol (see the example below).

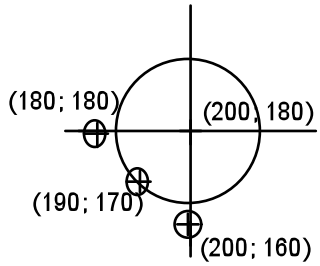
name: the name of the symbol being created;
group: selection of objects (points near the objects comprising the symbol);
coord: the origin of the symbol.

E.g.: Draw and create the symbol:



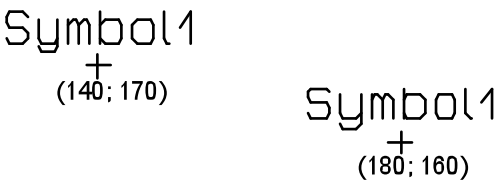
/SYMBDEF symbol1 180 180 200 160 190 170 ; 200 180

(ENTER (;) completes the selection)



You can place so-called “unresolved” symbols in the drawing (which are currently not in the symbol table); consequently, they are displayed in the drawing by their name:

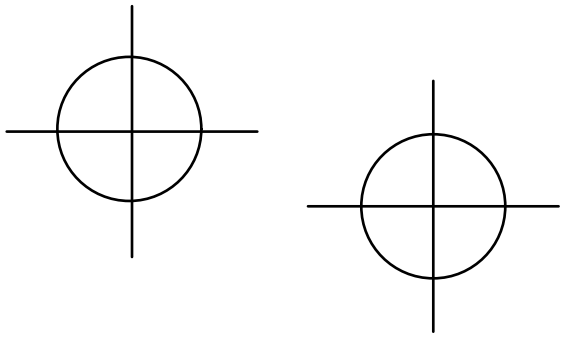
/POSITION /SYMBOL symbol1 140 170 170 120 ;



If you define the symbol, Symbol1, at a later time with the command:

/SYMBDEF symbol1 180 180 200 160 190 170 ; 200 180 ;

the command defines the symbol symbol1 and the defined symbol replaces the symbol names symbol1 in the drawing, i.e., resolves the unresolved symbol references:

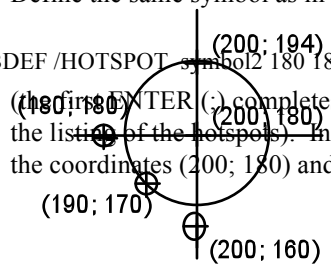


SYMBDEF HOTSPOT name group < coord >

Identical to the simple /SYMBDEF command as far as the command creates symbols comprising the selected objects. However, this command creates symbols with additional “handles”, so-called *hotspots*. One of the hotspots can be selected as origin. The symbol is placed by the current origin.

coord: coordinates of the origin (and optionally the hotspots) of the symbol.

E.g.: Define the same symbol as in the previous example, with two hotspots.



/SYMBDEF /HOTSPOT symbol2 180 180 200 160 190 170 ; 200 180 200 194 ;

(the first ENTER (;) completes the selection, the second one puts an end to the listing of the hotspots). In the example there are two hotspots defined at the coordinates (200; 180) and (200; 194).

Object Modifying Commands

ADJUST < single1 single2 >

Fits two objects together by shortening or extending them up to their intersection point nearest to the given point.
It retains the closer part of the objects to the pick point.

single1: a point near one of the objects;
single2: a point near the other object.

ADJUST FIRST < single1 single2 >

Fits the first object to the second one by shortening or extending it up to that intersection point which is closest to the pick points.
The command retains the closer part of the object to the pick point.

single1: a point near the object to adjust;
single2: a point near the object to which the first object will be adjusted.

ADJUST GROUP < single group >

Fits the selection of the objects against the first one. The longer part of each member of the selection will remain.

single: a point near the object to which the selected objects will be adjusted;
group: selection of the objects to adjust to the first one.

COMPLEMENT < single >

Creates the complementary part of a circular or elliptic arc and deletes the original one.

single: a point near the object to complement.

CUTTING < single1 single2 >

Cuts the two objects at their intersection point closest to the selection point.

single1: a point near one of the objects to cut;
single2: a point near the other object to cut.

CUTTING FIRST < single1 single2 >

Cuts the first selected object at the intersection point of the two objects closest to the selection point.

single1: a point near the objects to cut;
single2: a point near the cutting object.

CUTTING SECTION < coord1 coord2 < coord2 > >

Cuts a set of objects with a line given by its endpoints. The line is present only while entering the objects to be cut.

coord1: the starting point of the cutting section;
coord2: the endpoint of the cutting section;
coord2: points near the objects to cut. Enter completes a series of cutting and restarts the command. Giving Enter again completes the command

CUTTING { ALL | HERE } < single >

Cuts the object at all or the nearest intersection point(s) with other objects.

ALL: cuts the selected objects at all intersection points;
HERE: cuts the selected object at its intersection point nearest to the selection;
single: a point near the object to cut.

DELETE < single >

Deletes the chosen object. The deleted objects are revokable using the /UNDO command.

single: a point near the object to be deleted.

DELETE ALL

Deletes all existing objects (on all layers!). The deleted objects are revokable using the /UNDO command.

DELETE GROUP < group >

Deletes the selected group of objects. The deleted objects are revokable using the /UNDO command.

group: selection of the objects to be deleted.

DELETE PART < coord1 coord2 >

Deletes all items inside a window. Objects partially inside the window are cut and only the inner part will be deleted. The deleted objects are revokable using the /UNDO command.

coord1: a corner point of the window;
coord2: the other corner point of the window.

PDELETE < single >

Deletes the nearest part of the object to the chosen point, between the neighboring intersection points or endpoints of the object. The deleted objects are revokable using the /UNDOcommand.

single: a point near the object section to be deleted.

MODIFY attribute setting < group >

Changes the given attribute of the selected objects.

attribute: one of the keywords of attributes from the following list:

General attributes: COLOR, LTYPE, LWIDTH, LAYER

Text attributes: CHEIGHT, TDIRECTION, MFONT, CGAP, LGAP, CWIDTH, JUSTIFY, TORIGIN, CSLANT

Dim. attributes: DTCOLOR, DTLWIDTH, DTBOX, DTPOSITION, DTDIRECTION, DMCOLOR, DMLWIDTH, MKIND, MSIZE, DIGITS, DFORMAT, DTOLERANCE, ADIGITS, DAFORMAT, DATOLERANCE, PROJLINE, DSCALE

Hatch attributes: HDIRECTION, HSTEP, HOFFSET, HPATTERN

setting: the new value for the selected attribute;
group: selection of the objects whose attributes are to be modified.

E.g.: /MODIFY /LTYPE 2 /ALL /SLINE /EXCEPT /SLAYER 1_3 ;

Changes all lines to dash-line on all layers but layers 1, 2 and 3.

/MODIFY /COLOR 3 /SWINDOW 80 110 140 60 ;

Changes the color of lines to green inside a rectangle area.

MODIFY LTEXT text < single >

Links textual information to the selected object.

text: textual information to be part of a spreadsheet file;
single: a point near the object to be selected.

MODIFY GROUP < < attribute setting > group >

Changes all the selected attributes of the selected objects.

attribute: keywords of attributes;
setting: the new value for the selected attribute;
ENTER: completes selecting and changing the attributes;
group: selection of the objects whose attributes are to be modified.

MODIFY ENDPOINT < single coord >

Shifts the nearest end to the pick point of the selected line, polyline, or circular arc into the given point.

single: a point near the endpoint of the object to be shifted;
coord: the new position for the endpoint.

MODIFY LENGTH < single coord >

Modifies the length of the selected line, polyline, circular arc or a part of a dimension in such a manner that it moves the closer end of the object to the pick point along a construction line or arc in such a way that it should be in the closest position to the given point.

For arcs, a pick near the midpoint changes the radius with constant central angle; a pick near an endpoint moves the endpoint preserving the original radius.

single: a point near the endpoint of the object to move;
coord: a point defining the new position of the endpoint.

MODIFY NLENGTH < single length >

Modifies the length of the selected line, polyline, circular arc or a part of a dimension to the given numeric value in such a manner that it moves the closer end of the object to the pick point.

single: a point near the endpoint of the object to move;
length: the new length of the selected object.

MODIFY CONNECT << single > >

Connects the selected lines into a polyline using the same method as /ADJUST to fit the neighboring lines together.

single: points near the objects. Enter completes selecting a set of lines, connects them and starts with selecting another set of lines to be connected. Giving Enter again completes the command.

MODIFY { POLYLINE | SPLINE } ADDNODE < single coord >

Adds a new node to the selected polyline or spline between the nodes limiting the part closest to the chosen point.

POLYLINE: a new node will be added to the selected polyline part at the given point;
SPLINE: a new node will be added to the selected spline part at the given point;
single: a point near the object part the new node will be added to;
coord: the place for the new node.

MODIFY { POLYLINE | SPLINE } DELNODE < single >

Deletes the nearest node to the chosen point of the selected polyline or spline.

POLYLINE: the polyline node nearest to the given point will be deleted;
SPLINE: the spline node nearest to the given point will be deleted;
single: a point near the node to be deleted.

MODIFY { POLYLINE | SPLINE } MOVENODE < single coord >

Moves the selected node of the indicated polyline or spline to the given position.

POLYLINE: the polyline node nearest to the given point will be moved;
SPLINE: the spline node nearest to the given point will be moved;
single: a point near the node to move.
coord: the new place for the node.

MODIFY { POLYLINE | SPLINE } DELPART < single >

Deletes the nearest part to the chosen point of the indicated polyline or spline.

POLYLINE: the polyline part nearest to the given point will be deleted;
SPLINE: the spline part nearest to the given point will be deleted;
single: a point near the object part to delete.

MODIFY POLYLINE SMOOTH < single >

Replaces the polyline by a spline fitted to its nodes.

single: points near the polylines to smooth.

MODIFY POLYLINE SLICE < single >

Cuts the polyline into lines.

single: points near the polylines to convert.

MODIFY SPLINE { { OPENED | CYCLIC | ANTICYCLIC } < single>
I TANGENT < single { coord | VECTOR length angle } > }

- Modifies the type of spline to the given type, or
- changes the tangent of an opened spline at the selected end graphically or numerically.

OPENED: the type of the selected spline will be changed to OPENED;
CYCLIC: the type of the selected spline will be changed to CYCLIC;
ANTICYCLIC: the type of the selected spline will be changed to ANTICYCLIC;
single: a point near the spline to modify;
TANGENT: the tangent of the spline will be changed at the end nearer the selection;
single: a point near the spline whose tangent is to be changed;
coord: the point defining the new tangent graphically;
VECTOR: the tangent will be defined numerically;
length: the length of the vector defining the tangent;
angle: the angle of the vector defining the tangent.

MODIFY SPLINE ROUGH < single >

It replaces a spline with a polyline fitted to its nodes.

single: a point near the spline to convert.

MODIFY TEXT < single >

Replaces the pointed text by the *Preferred Text*.

single: a point near the text to replace.

EDIT single

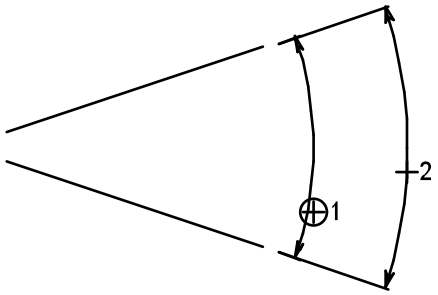
Opens a text editor and a window containing the selected text object. Leaving the editor the control returns to topCAD and the object will be replaced by the edited text.

single: a point near the text to be edited.

MODIFY DIMENSION < single coord >

Moves the dimension to a new place defined by the given point.

single: a point near any part of the dimension to move;
coord: a point defining the new position of the dimension.



MODIFY PDIMENSION < single coord >

Moves a part of a dimension to a new position.
If the selected part of the dimension is

- an *extension line*:
  the endpoint closest to the point to be dimensioned will move;
- a *dimension line*:
  the endpoint nearest to the given point will move on the original dimension line or circular arc of the dimension line to the nearest position to the given point;
- *text*:
  it will be dragged to the new position.

single: a point near the part of the dimension to move;
coord: a point defining the new position for the selected part of the dimension.

MODIFY DIMOPTION setting < group >

Changes the five dimension options of the selected objects.

setting: the new values for the five options;
group: selection of objects whose attributes are to be modified.

MODIFY DAUNIT { DECIMAL | DMS } < group >

Modifies the angle unit of some existing angle dimensioning to decimal degrees or degrees/minutes/seconds.

DECIMAL: sets the angle unit in the selected angle dimensions to decimal;
DMS: sets the angle unit in the selected angle dimensions to degrees/minutes/seconds;
group: selection of the angle dimensions to be modified.

MODIFY SYMBPAR < single num text >

Changes a textual symbol attribute ($1, ..., $8) of the selected symbol.

single: a point near the symbol whose attribute is to be changed;
num: the number of the symbol parameter to be changed (1 ...8);
text: the new (textual) value of the symbol parameter.

MODIFY SYMBNPAR < single num1 num2 >

Changes a numeric symbol attribute (#1, #2) of the selected symbol.

single: a point near the symbol whose attribute is to be changed;
num1: the number of the symbol parameter to be changed (1,2);
num2: the new (numerical) value of the symbol parameter.

MODIFY TOTAL < single < coord > >

Changes the geometrical characteristic of the selected objects.

single: a point near the object to be selected;
coord: coordinates of the modified object. (The number of coordinate pairs as well as the meaning of the new point depends on the type of the selected object. For example, in case of a circle the center point of the circle will be moved. In the case of a straight line its endpoints, i.e., the whole line, will be modified.)

OFFSET < length < group > ENTER >

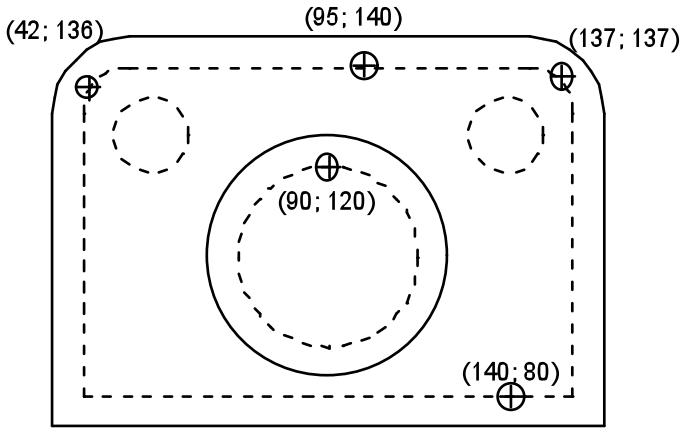
Offsets a set of closed chains parallel to themselves with the given distance. The chains must be entered object by object. Line, polyline, circular arc, and spline objects keep their type. Circles become circular arcs, ellipses and elliptic arcs become splines.

length: the offset distance;
group: selection of the objects in a chain to offset.

Selecting the objects in a chain must be completed with an ENTER, another ENTER must be given to complete the selection.

ENTER: prompts for another offset distance.

E.g.: /OFFSET 4 140 80 137 137 95 140 42 136 ; 90 120 ; ;



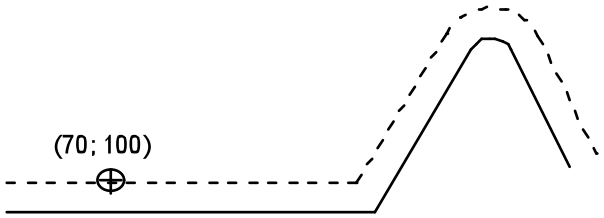
OFFSET OPENCHAIN < length < single > ENTER >

Offsets a set of open chains parallel to themselves with the given distance. The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines.
The whole chain or part of the chain can be selected:

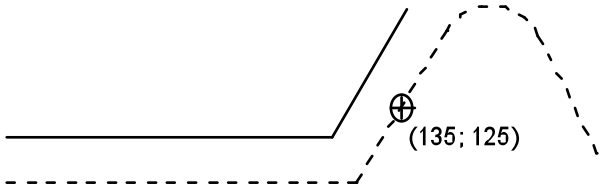
- ☐ pointing to the first or the last object in the chain nearer its free endpoint, the whole chain is selected;
- ☐ pointing to the first or the last object in the chain nearer its endpoint connected to the next object, only that object is selected;
- ☐ pointing to any other object in the chain, a part of the chain will be selected: from the object you pointed to, to that end of the chain which is farther from the selected point than from the midpoint of the segment of the chain nearest to the selection.

length: the offset distance;
single: a point near an object in the chain to offset
(selecting the chains must be completed with an ENTER);
ENTER: prompts for another offset distance, i.e., restarts the command.

E.g.: /OFFSET /OPENCHAIN 6 70 100 ; ;



/OFFSET /OPENCHAIN 6 135 125 ;

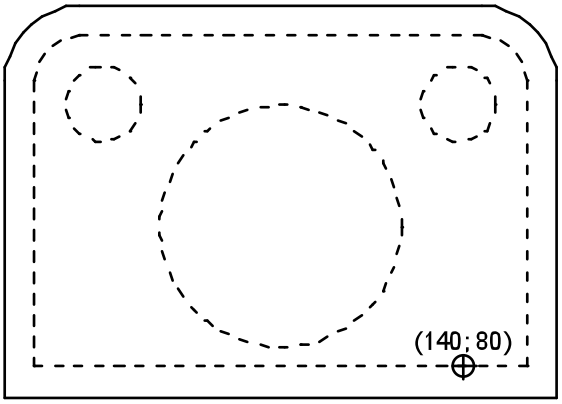


OFFSET CHAIN < length < single > ENTER >

Offsets a set of closed chains parallel to themselves with the given distance. A pick selects a complete chain. The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines.

length: the offset distance;
single: a point near an object in a chain to offset,
(selecting the chains must be completed with an ENTER);
ENTER: prompts for another offset distance, i.e., restarts the command.

E.g.: /OFFSET /CHAIN 6 140 80 ; ;



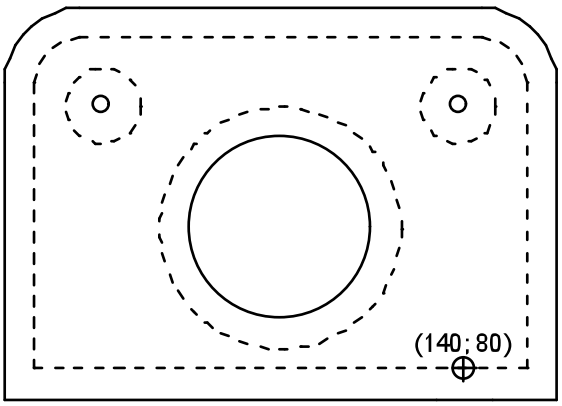
OFFSET INTCHAIN < length < single > ENTER >

Offsets a set of closed chains parallel to themselves with the given distance. A pick selects a complete chain with all its internal closed chains (only one level deep). The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines.

Notice that the internal chains are outlined in the opposite direction to the directly selected chains.

length: the offset distance;
single: a point near an object in a chain to offset,
(selecting the chains must be completed with an ENTER);
ENTER: prompts for another offset distance, i.e., restarts the command.

E.g.: /OFFSET /INTCHAIN 6 140 80 ; ;

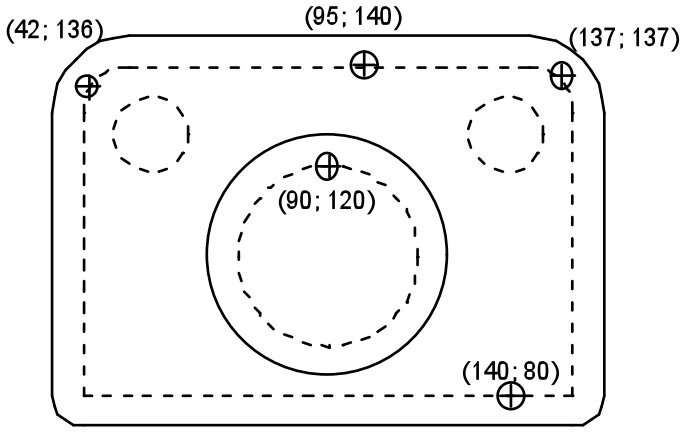


NC < length < group > ENTER >

Offsets a set of closed chains parallel to themselves exactly with the given distance at every point. The chains must be entered object by object. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines. If it is necessary, topCAD inserts circular arcs.

length: the offset distance;
group: selection of closed chains to be outlined,
(selecting the objects in a chain must be completed with an ENTER, another ENTER must be given to complete the selection);
ENTER: prompts for another offset distance.

E.g.: /NC 5 140 80 137 137 95 140 42 136 ; 90 120 ; ;



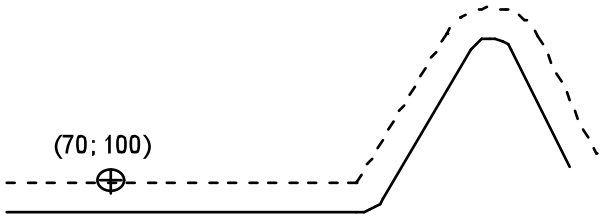
NC OPENCHAIN < length < single > ENTER >

Offsets a set of open chains parallel to themselves exactly with the given distance at every point. The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines. If it is necessary, topCAD inserts circular arcs.
The whole chain or part of the chain can be selected:

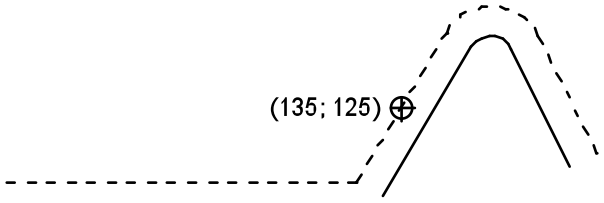
- pointing to the first or the last object in the chain nearer its free endpoint, the whole chain is selected;
- pointing to the first or the last object in the chain nearer its endpoint connected to the next object, only that object is selected;
- pointing to any other object in the chain, a part of the chain will be selected: from the object you pointed to, to that end of the chain which is farther from the selected point than from the midpoint of the segment of the chain nearest to the selection.

length: the offset distance;
single: a point near an object in the chain to offset
(selecting the chains must be completed with an ENTER);
ENTER: prompts for another offset distance i.e. restarts the command.

E.g.: /NC /OPENCHAIN 5 70 100 ; ;



/NC /OPENCHAIN 5 135 125 ; ;

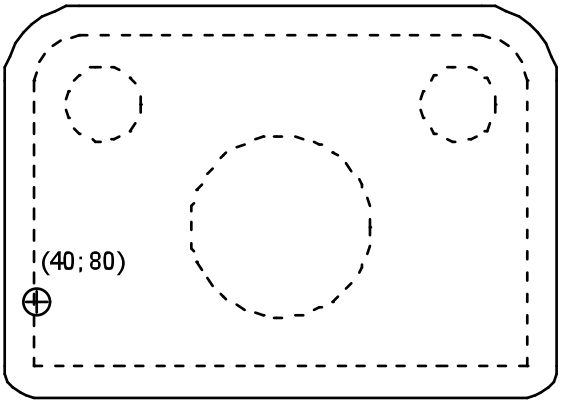


NC CHAIN < length < single > ENTER >

Offsets a set of closed chains parallel to itself exactly with the given distance at every point. A pick selects a complete chain. The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines. If it is necessary, topCAD inserts circular arcs.

length: the offset distance;
single: a point near an object in a chain to offset,
(selecting the chains must be completed with an ENTER);
ENTER: prompts for another offset distance i.e. restarts the command.

E.g.: /NC /CHAIN 5 40 80 ; ;

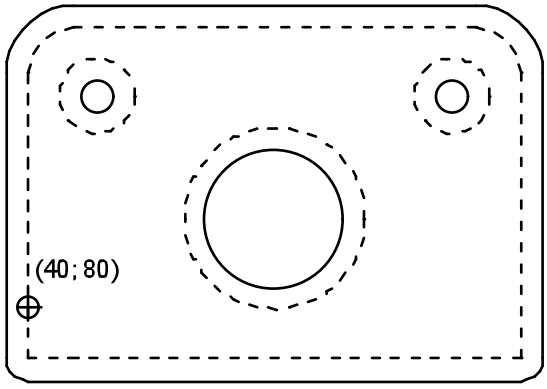


NC INTCHAIN < length < single > ENTER >

Offsets a set of closed chains parallel to itself exactly with the given distance at every point. A pick selects a complete chain with all its internal closed chains (only one level deep). The chains must be entered chain by chain. The lines, polylines, circular arcs, and splines keep their type, the circles become circular arcs, the ellipses and elliptic arcs become splines. If it is necessary, topCAD inserts circular arcs.
Notice that the internal chains are outlined in the opposite direction than the directly selected chains.

length: the offset distance;
single: a point near an object in a chain to offset,
(selecting the chains must be completed with an ENTER);
ENTER: It prompts for another offset distance, i.e., restarts the command.

E.g.: /NC /INTCHAIN 3 40 80 ; ;



UNDO BACKWARD num

Cancels the previous given number of operations.

num: number of operations to cancel.

UNDO FORWARD num

Revalidates the given number of canceled operations.

num: number of operations to revalidate.

UNDO { ON | OFF }

Switches the UNDO function on/off.

Move & Copy Commands

This group includes commands which move, stretch or replicate objects, group of objects, or a selected part of the drawing. The commands in this group use the *Preferred Transformation* and the *Preferred Repeat Factor*.

MOVE single

Moves an object using the *Preferred Transformation*.

single: a point near the object to be moved.

DUPLICATE single

Copies an object using the *Preferred Transformation*. It executes the operation "n" times where "n" is the current value of the *Preferred Repeat Factor*.

single: a point near the object to be duplicated.

MOVE GROUP group

Moves a selected group of objects using the *Preferred Transformation*.

group: selection of the objects to move.

DUPLICATE GROUP group

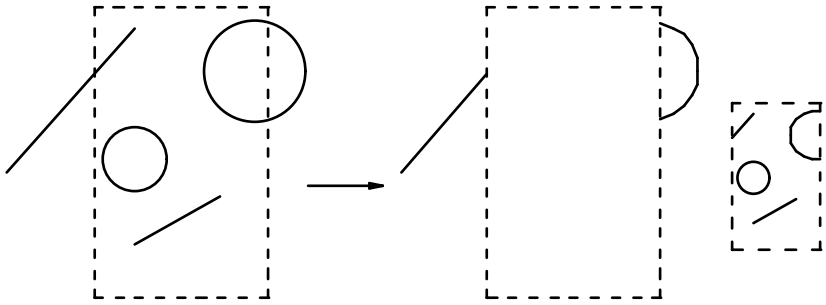
Copies a selected group of objects using the *Preferred Transformation*. It executes the operation "n" times where "n" is the current value of the *Preferred Repeat Factor*.

group: selection of the objects to copy.

MOVE PART [coord1 coord2]

Moves those object parts which fall inside a window using the *Preferred Transformation*.

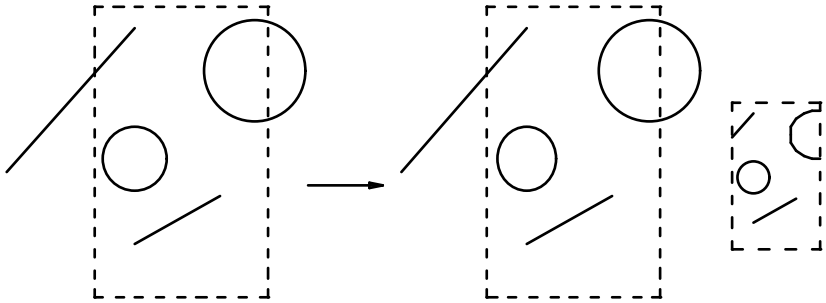
coord1: a corner of the window;
coord2: the opposite corner of the window.



DUPLICATE PART [coord1 coord2]

Copies those object parts which fall inside a window using the *Preferred Transformation*. This command does not use the *Preferred Repeat Factor*.

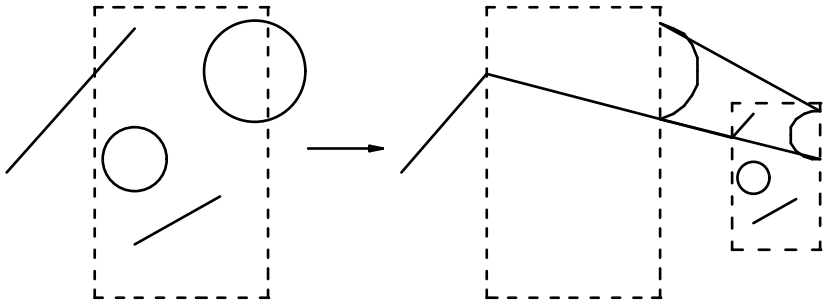
coord1: a corner of the window;
coord2: the opposite corner of the window.



MOVE CONTINUOUS [coord1 coord2]

Moves those object parts which fall inside a window using the *Preferred Transformation*, then it reconnects the breakpoints of the objects with lines.

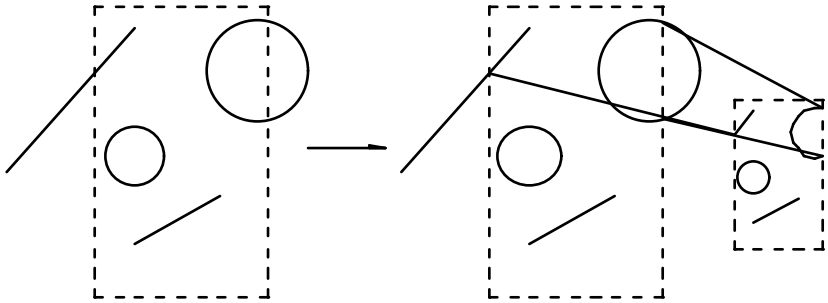
coord1: a corner of the window;
coord2: the opposite corner of the window.



DUPLICATE CONTINUOUS [coord1 coord2]

Copies those object parts which fall inside a window using the *Preferred Transformation*, than it reconnects the breakpoints of the objects with lines. This command does not use the *Preferred Repeat Factor*.

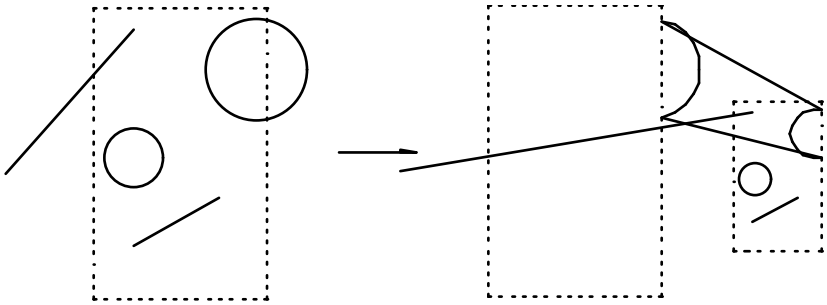
coord1: a corner of the window;
coord2: the opposite corner of the window.



STRETCH [coord1 coord2]

Moves those object parts which fall inside a window using the *Preferred Transformation*. If the object is a line or part of a polyline then topCAD transforms and moves only its endpoints and connects them with lines, otherwise it reconnects the breakpoints of the objects with lines.

coord1: a corner of the window;
coord2: the opposite corner of the window.



MLAYER layer < group >

Moves objects from the active layer to the given one.

layer: the target layer;
group: selection of the objects to move between layers.

CLEAR group

COPY [POSTSCRIPT] group

CUT [POSTSCRIPT] group

PASTE group

These commands execute the respective standard Macintosh command. The commands act on the selected group.

group: a selection of objects to be cleared, copied, cut or pasted.
POSTSCRIPT: the postscript description of the drawing will also be copied/cut into the Clipboard.

Query Commands

?SPACE set

Displays the space data about the selected layer(s) or all layers (number of objects per type, total number of objects and symbols, occupied area in bytes, the same data of revocable objects).

set: set of layers.

?ITEM [SPREADSHEET file]
{ < single > | SELECT [keyword] group }

Displays or puts into a spreadsheet file the data of the selected object(s), both general (serial number, type, sizes of comprising rectangle, layer, color, line type, line width) and specific data (endpoints of a line, length, direction, center point of a circle, radius etc.).

[SPREADSHEET file]: the name of the file the output will be put into (optional);
single: a point near the object whose data will be displayed;
SELECT: indicates that a set of objects will be selected;
keyword: modifier to further define or refine the selection;
group: selection of a set of objects.

E.g.: /?ITEM /SELECT /SWINDOW /IN 90 180 135 140 ;

Displays the data of objects inside a selection window. The window is determined by its opposite corners being at (90; 180) and (135; 140).

?COORDINATE < coord >

Displays the coordinates of the selected point(s).

coord: a point near the point to be selected.

?DISTANCE < coord1 coord2 >

Displays the distance and the horizontal and vertical projection between two points.

coord1: a point near one of the points;
coord2: a point near the other point.

?LENGTH < length >

Displays the selected length values.

length: a definition of the length to be displayed;

E.g.: /?LENGTH /PERIMETER 100 100 ;

The perimeter of the circle passing through the point (100; 100) will be displayed in the command window.

/VLENGTH L1 /?LENGTH /SIZE 100 100; /RECALL ;

The radius of the circle passing through the point (100; 100) will be assigned to the length type variable L1. In this command line the /?LENGTH command is nested in a variable definition command /VLENGTH. /RECALL re-calls the last used length-type value, that is the length of the object passing through the point (100; 100).

?ANGLE < angle >

Displays the given angle value(s).

angle: selects the angles to be displayed;

?GRID

Displays a window with the grid data (line type, color number, rarefaction factor, pen number, maximum number of lines, horizontal and vertical spacing).

?ACTUAL

Displays the current values of the *Preferred parameters* (used by certain object defining commands):

- ☐ *Preferred Radius*;
- ☐ *Preferred Half Axes* of an ellipse;
- ☐ *Preferred Angle*;
- ☐ *Preferred End Angles* of a circular arc or elliptic arc;
- ☐ *Preferred Repeat Factor*;
- ☐ parameters of the *Preferred Transformation*;
- ☐ translation vector;
- ☐ rotation angle;
- ☐ scale factors;
- ☐ *Preferred Text*.

?ATTRIBUTES

Displays a group of the current attributes:

- ☐ color;
- ☐ line type and line width;
- ☐ tolerance;
- ☐ layer;
- ☐ size of the previous, current and following window;
- ☐ distortion of the current window;
- ☐ character height and character width;
- ☐ character gap and line spacing;
- ☐ text direction;
- ☐ character slant;
- ☐ font (letter type);
- ☐ text origin;
- ☐ text justification;
- ☐ the type of the marker and its size;
- ☐ extension line distance from the marker;
- ☐ dimension scale factor;
- ☐ dimension text color and line width;
- ☐ the number of decimals of length and angle dimensions;
- ☐ the format of the dimension;
- ☐ the text of tolerances.

?SYMBOL [SPREADSHEET file] { TOP | DETAILED } { DEFINED | [SHORTLIST | WITHPARAMETER] group }

Displays information about symbols on the highest or all levels which are defined or are in the selected set of symbols.

SPREADSHEET: the data will be put into a file;
file: the name of the file the data will be put into;
TOP: the data of the symbols on the highest level will be displayed;
DETAILED: the data of symbols on all levels will be displayed;
DEFINED: the number of symbol occurrences will be displayed;
SHORTLIST: Lists the following data about the selected symbol:

- ☐ name
- ☐ symbol attributes with their values
- ☐ symbol origin
- ☐ translation vector
- ☐ rotation angle
- ☐ scale factors

WITHPARAMETER: lists the selected symbols;
group: selection of symbols whose data are to be listed.

?HATCH < single >

Displays the data of the hatched area (the type of hatching, perimeter, area, center of gravity, moments of inertia, direction of the main inertia axes).

single: a point near the hatched area.

?LAYER

Displays the number of Active, Modify Only (On) and Visible layers and the number of objects by layers.

?MENU

Displays the text of custom user icon commands, user defined menus, icons and custom tools.

?SNAP

Displays the coordinate snap grid parameters (distance between grid lines both vertically and horizontally) and the angle snap parameters (fixed snapping angles, classes of special points to which snapping is switched *on*).

?VARTAB

Displays the names of the defined simple variables and arrays, as well as the values of the simple variables and the dimensions of the arrays.

?UNDO

Displays a list of the last 20 steps which can be recalled by the /UNDO command.

?TOLERANCE

Displays the list of standard dimension tolerances available.

File Operations

This group includes commands in connection with handling files. The file definition used by the commands is of this form:

[[volume] < [:foldername] > :] filename

FOLDER { SFILE | DRAW | ENVIRONMENT | MACRO | PLOT | LIST } text

Sets the current folder for symbol files, drawings, environments, or macros.

SFILE: sets the current folder for a symbol file; or
DRAW: for drawing files; or
ENVIRONMENT: for an environment file; or
MACRO: for macro files;
PLOT: for plot files; or
LIST: for list files; or
text: the folder name.

OPEN { SFILE | DRAW | DXF | IGES | PICT | HPGL | ENVIRONMENT } { file | ENTER }

Loads the specified file into memory.

SFILE: the symbols which are currently in the memory into a symbol file;
DRAW: the current drawing into a drawing file in DXF, IGES, PICT or HPGL file format (or in topCAD's own file format as the default).
ENVIRONMENT: environment file (containing Preferred values, attribute values, user defined commands, etc.)
file: the name of the file to open;
ENTER: specifies the last used file to open.

MERGE { SFILE | DRAW } { ENTER | file }

Loads a symbol file or drawing without deleting the current symbols or drawing respectively. The drawing file is merged without transformation.

SFILE: symbol file
DRAW: drawing file
ENTER: merges the file of the last used filename
file: the name of the file to merge.

POSITION DRAW file < coord >

Positions the given drawing to the given location using the *Preferred Transformation*.

file: the name of the drawing file to be positioned;
coord: the point where the origin of the drawing will be positioned;
ENTER: completes positioning of drawings.

EDIT EFILE file

Opens an edit window with a text editor to edit a text file named file.
If the file already exists, it will be opened, if not, it will be created.
After quitting the editor control is returned to topCAD.

file: a file definition.

E.g.: /EDIT /EFILE ":Macro:myfile" ;

the file named *myfile* will be opened or created in the Macro folder.

PRINTFILE file

Opens the specified file for writing.

COMPRESS { YES | NO }

Compresses the data structures: clears the undo possibilities (normally used before a SAVE command).
The command needs a confirmation. If the answer is NO, the command has no effect.

E.g.: /COMPRESS ;

topCAD prompts for confirmation.
Entering YES, the command is executed.
Entering NO, the command is canceled.

/COMPRESS /YES ;

topCAD executes the command.

SAVE { SFILE | DRAW| DXF | IGES | PICT | ARCHICAD | HPGL | POSTSCRIPT | ASCII | ENVIRONMENT } { file | ENTER }

Saves the specified file into the current folder.

SFILE: puts the symbols which are currently defined (in the memory) into a symbol file;

DRAW: puts the current drawing into a drawing file in one of the following file formats (or in topCAD's own file format as the default):
DXF, IGES, PICT, ARCHICAD, HPGL, EPSF, MACRO

ENVIRONMENT: saves the current environment (Preferred values, attribute values, user defined commands, etc.);

file: the name of the file for saving;

ENTER: specifies the last used file.

Symbol Operations

POSITION SYMBOL [HOTSPOT num]symb < coord >

Positions the given symbol at the given locations using the *Preferred Transformation* and text parameters.

HOTSPOT: Instructs topCAD to use the defined hotspot as origin; if missing, the first defined hotspot will be used.
num: The sequence number of the hotspot to be used as origin
symb: The name of the symbol to be positioned
coord: The point the origin of the symbol will be positioned at

AUTONUMBER { coord1 coord2 | ENTER } { num1 num2 }
{ { LEFT | RIGHT } { BOTTOM | TOP | ENTER } |
{ BOTTOM | TOP } { LEFT | RIGHT | ENTER } | ENTER }
{ num3 | ENTER } {num4 | ENTER }
{ { LEFT | RIGHT } { BOTTOM | TOP | ENTER }
| { BOTTOM | TOP } { LEFT | RIGHT | ENTER } | ENTER }
{ num5 | ENTER } { num6 | ENTER } num7 group

Numbers the selected group of symbols within a selection rectangle. In numbering it uses the numeric parameter chosen. Giving Enter instead of an option keeps the previous setting.

coord1, coord2 opposite corners of the enclosing rectangular area;
ENTER means the whole area;
num1 number of boxes the area is divided into horizontally, and
num2 vertically;
LEFT, RIGHT, BOTTOM, TOP numbering of the boxes starts from left, right, bottom, or top;
num3 the initial number;
num4 the increment value between neighboring boxes;
LEFT, RIGHT,... the sort of the symbols within a box starts from;
num5 the initial symbol number;
num6 the increment value in numbering;
num7 the selected numeric parameter (#1 or #2);
group selection of the symbols to be numbered inside the rectangular area.

SEARCHLIB < file >

Selects a set of symbol files which will be searched for a symbol if the positioned symbol is not present in the memory.

file: Symbol file name.

CATALOGUE ENTER

Lists the current folders, files and autosave files for symbol libraries, drawings, macros, scan and list files, plotter files. It also lists the set of symbol libraries selected for searching.

CATALOGUE SYMBOL [DETAILED | ALL] { symb | ENTER }

Prepares a catalogue of the defined symbols (wildcards are allowed).

DETAILED: lists the defined symbols;
ALL: lists the defined symbols showing the number of comprising objects;
symb: the name of the symbol to be listed;
ENTER: all defined symbols will be listed.

CATALOGUE { SFILE | DRAW | ENVIRONMENT | MACRO } { ENTER | file }

Prepares a catalogue of the given type of files (wildcards are allowed).

ERASE SYMBOL { ENTER | symb } { YES | NO }

Deletes one or more symbols (wildcards are allowed). The command needs a confirmation. If the answer is NO, the command has no effect.

ENTER: deletes all defined symbols from the memory;
symb: the name of the symbol to be deleted;
YES: the symbol(s) will be deleted;
NO: the command will be canceled.

RENAME SYMBOL symb1 symb2

Renames a symbol.

symb1: the symbol name to be changed;
symb2: the new symbol name.

UNLINK < single >

Breaks down symbol, hatch, text or dimension-type objects into separate objects.

single: a point near the object to be unlinked;

UNLINK GROUP < group >

Breaks down a set of symbols, hatches, text or dimensions into separate objects.

group: the selection of objects to be unlinked;

PURGE { YES | NO }

Deletes all unused symbols from the memory. The command needs a confirmation. If the answer is NO, it has no effect.

RESOLVE { ENTER | file }

Attempts to resolve the unresolved symbol references using the given or the default symbol file.

ENTER: searches the default symbol file for the unresolved symbols;
file: the name of the file to search for the unresolved symbols.

ADDSYMB { ENTER | symb } { ENTER | file }

Adds one or more symbols into a symbol file.

ENTER: the last used symbol will be added;
symb: the name of the symbol to be added;
ENTER: the symbol will be added to the default symbol file;
file: the name of the file to which the symbol will be added.

Input/Output Device Control Commands

PEN num { PWIDTH length | set }

Assigns a plotter pen width or a set of logical colors to the num physical plotter pen.

num: a physical pen number;
PWIDTH: keyword: the assigned value will be interpreted as a penwidth;
length: value to be assigned as a penwidth;
set: a set of logical colors.

E.g.: /PEN 4 /PWIDTH 3;

The 4th pen will draw with a linewidth of 3.

/PEN 4 3_12;

The 4th pen will draw all objects with any color denoted by codes between 3 and 12.

PRINT scale [0|1|2] X offset Y offset { YES | NO } { YES | NO }

Makes an output to the printer set by the Chooser DA.

scale: Scale factor of the drawing (1:scale).
E.g.: the value 2 results in an output reduced by 2,
the value 0.5 results in an output enlarged by 2.
0,1,2: Selection.
0 means all objects in the drawing;
1 means objects appointed to by a window;
2 means objects selected currently.
X offset: Horizontal shift; the distance of the bottom-left corner of the smallest box around the drawing from the bottom-left corner of the drawing area on the sheet in x direction.
Y offset: Vertical shift; the distance of the bottom-left corner of the smallest box around the drawing from the bottom-left corner of the drawing area on the sheet in y direction.
first YES | NO: Rotation.
YES: rotates the drawing into vertical position (portrait),
NO: no rotation happens (landscape printing).
second YES | NO: Quality. Declares if Postscript output needed.

E.g.: /PRINT 2 /YES /YES

the current drawing is sent to the printer. The drawing is printed vertically on a Postscript printer. The lower left corner of the half reduced size drawing is shifted to the coordinates (30, 40).

PLOT { A0 | A1 | A2 | A3 | A4 | A5 | length1 length2 } { file | .AOUT | .BOUT } [scale] [X offset Y offset] [YES | NO]

Outputs the current drawing, at the specified scale, to a plotter with the specified paper size [length1 and length2 or ISO standard (A0...A5)], into a file, or directly to a port.

A0 through A5: standard sheet sizes;
length1: the horizontal size of a non-standard sheet;
length2: the vertical size of a non-standard sheet;
file: the name of the file you want to plot into (if you plot into a file instead of a plotter);
.AOUT: the output will be sent to the modem port;
.BOUT: the output will be sent to the printer port;
scale: scale factor (1:scale), E.g.:
the value 2 results in an output reduced by 2,
the value 0.5 results in an output enlarged by 2;
X offset: horizontal shift; the distance of the bottom-left corner of the smallest box around the drawing from the bottom-left corner of the drawing area on the sheet in x direction;
Y offset: vertical shift; the distance of the bottom-left corner of the smallest box around the drawing from the bottom-left corner of the drawing area on the sheet in y direction;
YES | NO: rotation:
YES: rotation into vertical position (portrait),
NO: no rotation (landscape).

E.g.: /PLOT 250 400 "gear " .2 20 30 /YES

makes a plot file of the current drawing in the file named “gear” with a sheet size of 250x400, enlarged by 5. The drawing will be rotated (portrait YES) and offset by 20 horizontally and by 30 vertically. The file will be put into the Plot folder.

TABLET { ON| OFF}

Defines whether a tablet is connected to the serial port of the computer.

E.g.: /TABLET /OFF

Disconnects the digitizer tablet.

EJECT

Executes the standard Macintosh function.

Formatted Input/Output

SCANF file < text1 < var > >

Reads a set of numerical or textual values from a file. Each line of the format text is assigned to a line of the scan file.

file: the name of the file;
text1: the format list is similar to that of the **printf** command in the "C" language (it can be a multiline text):

%d means an integer value;
%f means a floating point value;
%x means a hexadecimal integer value;
%o means an octal integer value;
%c means a character;
%s for a textual value from the current position to the next space " ";
%l means a textual value from the current position to the end of line.

var: variable to which the value read from the file will be assigned.

SCANFILE file

Opens the specified file for reading.

PRINTF file < text1 < { num | text2 } > >

Outputs a set of numerical or textual values into a file.

file: the name of the file;
text1: the format list (it can be a multiline text):

%d means an integer value,
%f means a floating point value;
%x means a hexadecimal integer value;
%o means an octal integer value;
%c means a character;
%s for a textual value from the current position to the next space " ";
%l means a textual value from the current position to the end of line.

num: numerical value to be written into the file;
text2: textual value to be written into the file.

SPRINT var < text1 < num I text2 >>

Writes a set of formatted numerical or textual values into a text type variable. The variable must already have been defined.

var: an existing, text type variable;

text1: the format list is the same as in the/PRINTF and /SCANF commands and similar to that of **printf** in the "C" language. (It can be a multiline text.)

%d means an integer value;
%f means a floating point value;
%x means a hexadecimal integer value;
%o means an octal integer value;
%c means a character;
%s for a textual value from the current position to the next space " ";
%l means a textual value from the current position to the end of line.

num: numerical value to be written into the variable;

text2: textual value to be written into the variable.

Example: We want to write some variables into a text type variable.

1. Let us define some variables.
   /vint i 11
   /vfloat f 456.128
   /vtext t "abcde"

   These variables will be written into text type variables.

2. Let us define some text variables.
   /global /vtext apple1 " "
   /global /vtext apple2 " "
   /global /vtext apple3 " "
   /global /vtext apple4 " "

3. Let us write the variables i f and t into apple1 without any formatting.
   /sprint apple1 "%d %f %s" (i) (f) /use t ;

   The value of the variable apple1 is:
   "11 456.127991 abcde"

   The variables are written into apple1 with a space character between them.

4. Let us write the variables into apple2 with formatting.
   /sprint apple2 "%5d %8.2f %7s" (i) (f) /use t ;

   The value of the variable apple2 is:
   “ 11 456.13 abcde”

   In the format list

"%d"	Means an integer value
"%5d"	Means that we define a length of 5 characters for the integer value, justified to the right
"%d-5"	Means the same, but justified to the left
"%8.2f"	Means a floating point value with a total length of 8 characters with 2 decimals justified to the right. (A negative value would mean justification to the left.)

5. Let us write the variables into apple3 with formatting and some character constants.
   /sprint apple3 "integer:%5d floating point:%8.2f text:%7s" (i) (f) /use t ;

   The value of the variable apple3 is:
   “integer: 11 floating point: 456.13 text: abcde”

6. Let us write the variables into apple4 with formatting and character constants, but justified to the left.
   /sprint apple4 "integer:%-5d floating point:%-8.2f text:%-7s" (i) (f) /use t ;

   The value of the variable apple4 is:
   “integer:11 floating point:456.13 text:abcde “

7. Finally, let us compare the results to see the effect of different format definitions.

apple1:	“11 456.127991 abcde “
apple2:	“ 11 456.13 abcde”
apple3:	“integer: 11 floating point: 456.13 text: abcde“
apple4:	“integer:11 floating point:456.13 text:abcde “

LISTFILE { ON | OFF }

Following a /LISTFILE /ON command, the output of the /LIST command and the query commands (starting with "?") are put into a file in the List folder until the next /LISTFILE /OFF command. The next /LISTFILE /ON command opens the next file. The files are named topList0, topList1, ...

LIST < text1 < { num | text2 } > >

Lists the specified variables in the command window or into a file if /LISTFILE /ON has been given.

text1: the format list is similar to that of the **printf** command in the "C" language (it can be a multiline text):

%d means an integer value, it must be given between " " delimiters;
%f means a floating point value;
%s means a textual value.

num: numerical value to be written into the file;
text2: textual value to be written into the file.

E.g.: /LIST "%f %f" (d) (k)
/LIST "%d %d" (array1[3][1][1]) (array[4][1][1])

CLOSEFILE file

Closes the specified file.

Attributes & Preferred Values Definitions

This group contains commands which are used to set general parameters, different kind of object attributes as well as preferred values for some object creating commands.

Basic Settings

AUNIT num

Sets the Input Scale factor.

TOLERANCE num

Sets the radius of the tolerance circle in relative tolerance units. The higher num is given the greater the tolerance circle will be.

UNIT { MM | CM | M | { INCH | FEET } num }

Sets the default drawing unit. All numerical values of length and coordinates will be interpreted and displayed in this unit. In case of inch and feet the precision must also be given.

mm, cm, ..: The drawing unit is set to mm, cm,...
num: precision of the unit when the drawing unit is inch or foot. The acceptable values are:

1:	means	1/1"
2:		1/2"
4:		1/4"
.		.
.		.
128:		1/128"

AUNIT { DECIMAL | DMS }

Sets the default angle unit. All angle values and dimensions will be displayed in this unit.

DECIMAL: angle dimensions (fractions of degrees) in decimal units;
DMS: angle dimensions (fractions of degrees) in minutes and seconds.

SNAP { length1 { ENTER | length2 { ENTER | length3 { ENTER | length4 } } } | ON | OFF | ALTERNATE | GRID num }

Sets the parameters of the snap grid or chooses one of the grids to be used as a snap grid.

length1: snap grid horizontal step;
first ENTER: keeps the snap grid vertical step;
length2: snap grid vertical step;
second ENTER: keeps the snap grid horizontal offset;
length3: snap grid horizontal offset;
third ENTER: keeps the snap grid vertical offset;
length4: snap grid vertical offset;
ON: switches on the snap function;
OFF: switches off the snap function;
ALTERNATE: toggles the snap function, i.e., switches it on if it is off and vice versa;
GRID: uses a grid as a snap grid;
num: the number of the grid to be used as a snap grid.

ASNAP { < angle > | HV | OFF | ON | ALTERNATE }

Sets up to eight snap angles or switches on/off the angle snap.

angle: a snap angle;
HV: sets the snap angles to 0˚, 90˚, 180˚, 270˚;
OFF: switches off;
ON: switches on the angle snap function;
ALTERNATE: toggles the angle snap function, i.e., switches it on if it is off and vice versa.

FSPECIAL { ON | OFF } { ALL | ENTER | < { EXTREMUM | CENTER | MIDDLE | FOCUS | INTERSECTION | NEAREST } > { ENTER | SSYMBOL | SHATCH } ENTER

Switches on/off automatic gravity for the selected type of special point. Automatic gravity means that if a special point falls within the pointer tolerance radius, topCAD uses the coordinates of that point as the coordinates for a new point. Gravity can be switched off separately for symbols and hatches.

ON: switches on the function;
OFF: switches off the function;
ALL: selects all classes of special points;
ENTER: the last selected class of special points remains valid;

EXTREMUM, CENTER, ...: restricts the selected class of special points to extremum, center point, and so on;
ENTER: finishes defining special point classes;
SSYMBOL: defines automatic gravity for symbols;
SHATCH: defines automatic gravity for hatches;
ENTER: completes the command.

ICURSOR { ON | OFF | ALTERNATE }

Switches on/off the automatic snapping to special points. If the "intelligent cursor" is switched on, the pointer snaps to special points falling within the tolerance circle around the pointer while changing shape according to the kind of special point.

ON: Turns the automatic snapping on;
OFF: turns the automatic snapping off;
ALTERNATE: toggles the automatic snapping, i.e., turns the function on if it is off and vice versa.

NODES { ON | OFF | ALTERNATE } { ALL | ENTER | < { SSPLINE | SPOLYLINE | STEXT | SSYMBOL | SCIRCLE | SCARC | SELLIPSE | SEARC | SHATCH | SLINE | SDIMENSION } > ENTER }

Switches on/off displaying the markers of the nodes or the special points of the selected class of objects.

ON: switches on displaying;
OFF: switches off displaying;
ALTERNATE: toggles displaying the nodes and special points, i.e., turns the function on if it is off and vice versa;
ALL: selects all classes of objects;
ENTER: the last selected class of objects remains valid;
SSPLINE, SPOLYLINE,...: restricts the selected class of objects to splines, polylines, and so on;
ENTER: completes the command.

FTEXT { ON | OFF }

Switches on/off the fast text drawing mode. With fast text drawing switched ON, all text is replaced by rectangular frames.

TMIRROR { ON | OFF }

Determines whether mirroring a selection with text in it mirrors the text or not.

TXREADABLE { ON | OFF }

Determines whether text remains the right way up or not when transforming (rotating) a selection with text in it.

E.g.: /TXREADABLE /ON

Without this command, if you rotated, say, a symbol by 180˚ symbol text would be turned upside down. Switching this option ON, only the rectangle enclosing the text is rotated. The text itself remains the right way up.

TLAYER { ACTIVE | ORIGINAL }

Determines whether copied objects are assigned to

ACTIVE: the ACTIVE layer; or
ORIGINAL: the original layer (on which the objects exist).

AUTOSAVE { TIME num | STEP num | OFF | ON | ALTERNATE }

Specifies whether a temporary backup of the drawing should be automatically saved after a given time period or number of operations.

TIME: automatic save after every num seconds;
num: the time in seconds between two consecutive automatic saves;
STEP: automatic save after every num operations;
num: the number of operations between two consecutive automatic saves;
OFF: turns off the automatic saving;
ON: turns on the automatic saving;
ALTERNATE: toggles the automatic save function, i.e., turns it on if it is off and vice versa.

JOURNAL { ON | OFF | ALTERNATE }

Turns on or off the journal file recording.

OFF: turns off the journal file recording;
ON: turns on the journal file recording;
ALTERNATE: toggles the journal file recording, i.e., turns it on if it is off and vice versa.

FDATE num

Sets the format of date throughout topCAD.

num: Identifier of date format (1-4).

1: 6:25 PM	2: 6:25:10 PM
3: 18:25	4: 18:25:10

FTIME num

Sets how time string should appear in topCAD.

num: Identifier of date format (1-6).

1: 20/03/92	2: 03.20.92
3: 92.03.20	3: 20 Mar 1992
5: Fri, 20 Mar, 1992	6: Friday, 20 March, 1992

PRNOTES text

Lets you add notes on your project.

text: textual information to be added.

PRSUMMARY < num text >

Adds pieces of summary infomation on your project.

num: field identifier in the summary table (1-9).

1 Project description
2 Subject
3 Drawing number
4 Job number
5 Chief engineer
6 Designer
7 Supervisor
8 Note
9 Date

text: textual information to be added.

Attribute Defining Commands

GLOBAL attribute_setting

Defines the following attribute settings as PERMANENT inside a main command. Attribute setting is otherwise TEMPORARY, if defined inside a main command.

attribute_setting: one of the appropriate attribute defining commands.

E.g.:
```
/COLOR 1
/CIRCLE /CPOINT /COLOR 2  100 100  200 100  ;
/LINE /SINGLE  100 100  200 100  ;
```

The first command line sets the current color to black. A red circle is drawn by the next line. Because the color setting is temporary, the section drawn by the third command line will be black.

```
/COLOR 1
/CIRCLE /CPOINT /GLOBAL /COLOR 2  100 100  200 100  ;
/LINE /SINGLE  100 100  200 100  ;
```

This command sequence draws a red circle, and a *red* line.
It is because of the /GLOBAL command which sets the color attribute permanently. (The first command /COLOR 1 no longer has an effect.)

ATTRIBUTES DEFAULT { ALL | ENTER | GENATTR | TEXTATTR | DIMATTR }

Sets all attributes, or only the general, text or dimension attributes to the default values.

LIKE single

Sets the current attributes to the attribute values of the selected object.

single: a point near the object.

COLOR { LIKE single | num }

Sets the current logical color number:

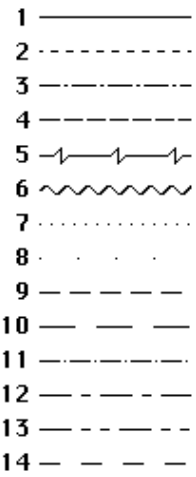
LIKE: choosing the color of an existing object;
single: a point near the object; or
num: a logical color number.

LTYPE { LIKE single | num }

Sets the current line type:

LIKE: choosing the line type of an existing object;
single: a point near the object; or
num: the number of a line type.

The line type codes:



15-128: are user defined line types (see *Defining a custom line type* at the description of the *Customization.../Line Types...* command in the **User's Guide**).

LWIDTH { LIKE single | num | FINE | NORMAL }

Sets the current line width:

LIKE: choosing the line width of an existing object;
single: a point near the object; or
num: the numerical value of a line width
(the input is in the default length unit); or

switches the drawing of wide lines between

FINE: draws rounded ends (this takes more time to draw), and
NORMAL: draws square ends.

PRECISION { LIKE single | num }

Sets the current accuracy of splines.

LIKE: choosing the accuracy of an existing spline;
single: a point near the spline; or
num: a value for the accuracy.

CHEIGHT { LIKE single | length }

Sets the current character height for text and dimension text (custom and topCAD fonts).

LIKE: choosing the character height of an existing object;
single: a point near to the object; or
length: the value for the character height.

CWIDTH { LIKE single | num | PROPORTIONAL | FIXWIDTH }

Sets the character width for custom and topCAD fonts:

LIKE: choosing the character width of an existing object;
single: a point near to the object; or
num: the character width in percent of character height;
PROPORTIONAL: each character has its individual width;
FIXWIDTH: all characters have the same width.

CGAP { LIKE single | num }

Sets the gap between characters:

LIKE: choosing the gap of an existing object;
num: a numerical value for the gap in percent of character height.

LGAP { LIKE single | num }

Sets the gap between the text lines:

LIKE: choosing the gap of an existing object;
single: a point near to the object; or
num: a numerical value for the gap in percent of character height.

CFONT { LIKE single | num }

Sets the current font. The purpose of this command is as that of /MFONT but it uses another numbering to identify fonts. This command is for compatibility with earlier versions of topCAD.

LIKE: choosing the font of an existing text or dimension text;
single: a point near the text; or
num: a numerical value for the font number in the range 0-255. Font number 0 means the topCAD font, values 1–16 mean Macintosh fonts, values above 16 are reserved for custom fonts.

MFONT { LIKE single | num }

Another way to set the current font, setting by its Macintosh font–id number:

LIKE: choosing the font of an existing text or dimension text;
single: a point near the text; or
num: a numerical value for the font number. The font–id is a unique identifier. A value in the range -32768 – +32767 identifies a Macintosh font. If the font exists in the System folder, the font itself is used; if missing, the system font is used instead. topCAD fonts and user fonts (so called vector fonts) are identified by the numbers num≥100.000.

TDIRECTION { LIKE single | angle }

Sets the current text direction:

LIKE: choosing the direction of an existing text or dimension text;
single: a point near the text; or
angle: an angle value for the text direction.

CSLANT { LIKE single | angle }

Sets the current character slant relative to text direction:

LIKE: choosing the slant of an existing text or dimension text;
single: a point near the object; or
angle: an angle value for the character slant in the range 15° -165°.

TORIGIN { LIKE single | num }

Selects the text origin:

LIKE: choosing the origin of an existing text;
single: a point near the text; or
num: a value specifying the text origin:

1: lower left corner,
2: mid-left,
3: upper left corner,
4: mid-bottom,
5: center,
6: mid-top,
7: lower right corner,
8: mid-right, or
9: upper right corner.

JUSTIFY { LIKE single | LEFT | CENTERED | RIGHT }

Sets the current justification for multiline text:

LIKE: choosing the justification of an existing text;
single: a point near the text; or
LEFT: selects left justification;
CENTERED: selects centered justification;
RIGHT: selects right justification.

HOFFSET { LIKE single | length }

Sets the current hatch offset (distance between the hatch lines or symbols):

LIKE: choosing the offset of an existing hatch;
single: a point near the hatch; or
length: a value for the offset.

HSTEP { LIKE single | length }

Sets the current hatch step (distance between the hatching symbols):

LIKE: choosing the step of an existing symbol hatch;
single: a point near the symbol hatch; or
length: a value for the step.

HDIRECTION { LIKE single | angle }

Sets the current hatch direction (direction of hatch lines or columns of symbols):

LIKE: choosing the direction of an existing hatch;
single: a point near the hatch; or
angle: a value for the direction.

DIGITS { LIKE single | num }

Sets the current number of decimal digits in the dimension numbers:

LIKE: choosing the decimal digit number of an existing dimension;
single: a point near the dimension; or
num: a value for the decimal digit number.

Negative values lead to the suppression of displaying the zero value decimals.

ADIGITS { LIKE single | num }

Sets the current decimal digit number in the angle quotations:

LIKE: choosing the decimal digit number of an existing angle dimension;
single: a point near the angle dimension; or
num: a value for the decimal digit number.

Negative values lead to the suppression of displaying the zero value decimals.

DSCALE { LIKE single | num }

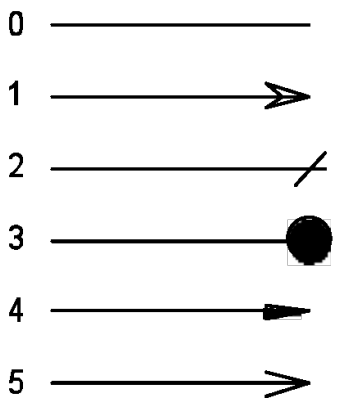
Sets the current scale factor for dimensions:

LIKE: choosing the scale factor of an existing dimension;
single: a point near the dimension; or
num: a numerical value for the scale factor.

MKIND { LIKE single | num }

Selects the current marker type at the ends of a dimension line:

LIKE: choosing the marker type of an existing dimension;
single: a point near the dimension; or
num: a numerical value for the marker type ranging from 0 through 5:



MSIZE { LIKE single | length }

Sets the size of the markers at the ends of the dimension lines:

LIKE: choosing the marker size of an existing dimension;
single: a point near the dimension; or
length: a numerical value for the marker size. Negative values lead to mirroring of the arrows by the extension line.

DIMOPTION num1 num2 num3 num4 num5

Sets the current values of the dimension options. See the *Dimension...* command on the Attributes menu in the User's Guide.

num1, num2,... values for the five dimension option. The options are numbered from left to right and from the top downwards.

DTCOLOR { LIKE single | num }

Sets the logical color number for dimension text:

LIKE: choosing the logical color number of an existing object;
single: a point near the chosen dimension; or
num: a numerical value in the range of 0...255.

DTLWIDTH { LIKE single | length }

Sets the line width for dimension text:

LIKE: choosing the line width of an existing object;
single: a point near the chosen dimension; or
length: a numerical value in the range of 0...25.5 mm.
(Note that the input is in the default length unit).

DTBOX { LIKE single | num }

Framing of dimension text.

LIKE: choosing the appearance of an existing dimension text;
single: a point near a dimension-type object;
num: 0 or 1. Enter 1 to have the dimension text framed.

DTPOSITION { LIKE single | num }

Setting the current position of dimension text above, on or below the dimension line.

LIKE: choose the position of an existing dimension text object;
single: a point near a dimension-type object;
num: a numerical value between 0 and 2, meaning above, on, and below, in this order.

DTDIRECTION { LIKE single | num }

Setting the current direction of dimension text to horizontal, vertical, parallel, or standard.

LIKE: choose the direction of an existing dimension text object;
single: a point near a dimension-type object.
num: a numerical value between 0 and 3, meaning horizontal, vertical, parallel, and standard, in this order.

DMCOLOR { LIKE single | num }

Setting the current color of the dimension marker.

LIKE: choose the color of an existing dimension marker;
single: a point near a dimension-type object;
num: color.

DMLWIDTH { LIKE single | num }

Setting the current line width of the dimension marker.

LIKE: choose the width of an existing dimension marker;
single: a point near a dimension-type object;
num: line width.

DFORMAT { LIKE single | text }

Sets the format of the dimension text:

LIKE: choosing the format of an existing dimension;
single: a point near the dimension; or
text: a text to be part of the dimension text.

The "#" character represents the measured value, the "^" character means the symbolic value of the UNI tolerance if it is present.

E.g.: /DFORMAT "Diameter = # mm";

The dimension text will be "Diameter = 10.00 mm"
(the dimensioned size is 10.00 mm).

DAFORMAT { LIKE single | text }

Sets the format of angle quotations:

LIKE: choosing the format of an existing angle dimension;
single: a point near the dimension; or
text: a text to be part of the dimension text.

The "#" character represents the measured value.

DTOLERANCE { ENTER | { LIKE single1 | text1 } { ENTER | LIKE single2 | text2 } }

Sets the text of tolerances of the length quotations. Both tolerances (upper and lower) can inherit the tolerance text of an existing dimension or can be entered as a text. ENTER means both tolerance texts or the lower one will be cleared. If the first text is identical to one of a standard tolerance and the second one is empty, the program will replace them with the correct computed values.

ENTER: resets tolerance format (no tolerance values displayed in length dimensions);
LIKE: defines the format of tolerance as that of an existing dimension;
single1: a point near the object with the tolerance format to be picked up;
text1: upper tolerance text;

ENTER: resets tolerance format to display no lower tolerance values in length dimensions;
LIKE: defines the format of lower tolerance as that of an existing dimension;
single2: a point near the object with the tolerance format to be picked up;
text2: lower tolerance text.

DATOLERANCE { ENTER | { LIKE single1 | text1 } { ENTER | LIKE single2 | text2 } }

Selects the text of tolerances of angle quotations. Both tolerances (upper and lower) can inherit the tolerance text of an existing dimension or can be entered as a text. ENTER means both tolerance texts or the lower one will be cleared. See /DTOLERANCE above.

PROJLINE { LIKE single | length | NO }

Sets the length of the projection lines

LIKE: choosing the line width number of an existing object;
single: a point near the dimension with the extension line width to be picked up; or
length: length≥0:
projection lines start from the point to be dimensioned,
length<0:
projection lines start from the marker;
NO: no projection line will be drawn.

SYMBPAR num text

Assigns the given text as a value to the *$num* textual symbol parameter.

num: number of the symbol attribute (1 ... 8);
text: the text value for the attribute.

SYMBPAR { ON | FORMAL | OFF }

Determines whether the text attributes of symbols appear on the screen or not. The attributes are displayed

ON: by their values;
FORMAL: by their names ($1 ... $8);
OFF: they will not appear in the drawing.

SYMBNPAR num1 num2

Assigns the given number as a value to the *#num* numerical symbol attribute.

num1: the number of the symbol attribute (1 or 2);
num2: value for the attribute.

SYMBNPAR { ON | FORMAL | OFF }

Determines whether the numerical attributes of symbols appear on the screen or not. The attributes are displayed

ON: by their values;
FORMAL: by their names (#1, #2);
OFF: they will not appear in the drawing.

SYMBTEXT text

Assigns the specified text as a comment for symbols.

Preferred Values Defining Commands

{ ABEGIN | AEND } angle

They set the *Preferred End Angles*.

ANGLE angle

Sets the *Preferred Direction*.

ATEXT text

Sets the *Preferred Text*.

E.g.: /ATEXT shaft1
/ATEXT "This is the Preferred Text"

If there are spaces or special characters in the text, the quotation marks " " are obligatory.

RADIUS length

Sets the *Preferred Radius*.

E.g.: /RADIUS (/topradius + 17)
(topradius is the topCAD variable for the *Preferred Radius*)

MAJOR length

Sets the *Preferred Major Half Axis*.

MINOR length

Sets the *Preferred Minor Half Axis*.

REPEAT num

Sets the *Preferred Repeat Factor*.

AXFORM { ON | OFF | XCENTER | tran }

Sets the *Preferred Transformation*, turns the transformation on or off, or switches on the transformation of the center point of the object. All measurements are made from the drawing origin (in absolute coordinates).

ON: switches on the current transformation
OFF: switches off (ignores) the current transformation
XCENTER: switches on the transformation of the center point of the object
tran: sets the *Preferred Transformation* in the absolute system of coordinates (the validity of this setting is PERMANENT).

BCOLOR num

Sets the background color.

num: logical color number (0...255).

SORIGIN coord

Sets the new drawing origin.

coord: coordinates where the new drawing origin (the zero point) should be put.

Set Defining Commands

It is the commands that manage *Attribute* and *Preferred Values* sets belong to this group. These commands save/restore the current attributes and preferred values.

ATTRIBUTES { PUT | GET } { ALL | ENTER | GENATTR | TEXTATTR | DIMATTR } { STACK | num }

Stores (PUT) or restores (GET) the general, text, dimension, hatch and symbol attributes and the *Preferred Values* into/from stack or the selected set (0...9).

PUT: stores the current attributes and/or *Preferred Values*;
GET: restores the current attributes and/or *Preferred Values*;
ALL: stores/restores all kind of attributes and the *Preferred Values*;
ENTER: stores/restores the *General Attributes* and the *Preferred Values*;
GENATTR, TEXTATTR, DIMATTR: stores/restores the *General Attributes* , or text or dimension attributes alone;
STACK: stores/restores the values using stack instead of an attribute and preferred value set;
num: the number of the attribute and preferred value set to be used.

DEFATTRIBUTES { SSPLINE | SPOLYLINE | STEXT | SSYMBOL | SCIRCLE | SCARC | SELLIPSE | SEARC | SHATCH | SLINE | SDIMENSION | SPOINT } num

Selects a set of attributes (0, ..., 9) to get the current attributes from. These attributes will be assigned to the objects created following this command . If the numerical value is -1, the attributes will not be changed.

SSPLINE, SPOLYLINE, ...: selects the class of objects the relevant attributes of the set should be applied to;
OFF: switches off the temporary transformation in its own system of coordinates;
num: identifies the set by number.

Customization Commands

The commands in this group serve to define custom fonts and custom line types which can be referenced in the same way as any other fonts or line types. See the /LTYPE and /CFONT commands earlier in this chapter.

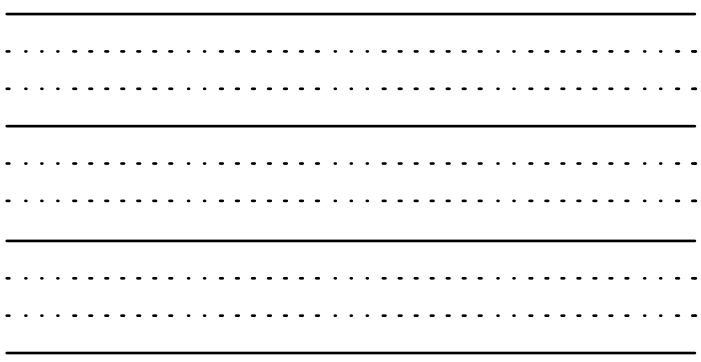
COMPOSITE name < num1 num2 num3 num4 num5 num6 >

Defines a new custom composite hatch type. Up to four different line type components can be combined into the composite hatch type. Line type, line width, angle, distance between two neighbouring line of the same type and periodicity can be defined group by group.

name: name of the composite hatch;
num1: line type of the line components;
num2: line width of the line components;
num3: distance between two parallel lines;
num4: angle of the line components;
num5: number of periods;
num6: binary value to define which lines in period should be drawn, which to be omitted. 1 digit stands for drawing the line, 0 for omitting.

E.g.: /composite insulation 1 0 5 0 3 1 7 0 5 0 3 6 ; ;

defines a pattern made of two different line types. The angle of both line types is 0°. Each parallel line is drawn by 5 mms. A period consists of three lines. In the first group, which consists of contiguous lines, the first two lines are omitted, the third is drawn ($001_2 = 1_{10}$). In the second group, which consists of dotted lines, the first two lines are drawn, the third one is omitted ($110_2 = 6_{10}$).



FONTDEF num < char [BOUNDS num1 num2 num3 num4] group >

Defines a character of the user defined font numbered num. The character bounds can also be set.

num: the font number >100.000;
char: the original character of the given font representing the defined character;
BOUNDS: the boundary of the defined character will be specified (the normal boundary is a 100*100 rectangle);
num1: top limit (1 through 150);
num2: bottom limit (-50 through 99);
num3: right limit (1 through 150);
num4: left limit (-50 through 99);
group: selects the object representing the character.

ULINETYPE num1 < num >

Defines a new custom line type.

num1: identifies the custom line type by number (15...128);
num: values for line or gap lengths expressed in relative units (max. 8 values).

E.g.: /ULINETYPE 22 1 2 3 4 5 6 7 8 ;
defines the custom line type 9. The length of the first section is 1 unit, the length of the gap next to it is 2, and so on.

Parameter Definitions

These commands can be used when topCAD is waiting for a parameter, such as a point, length, angle, and so on. When the syntax of a command requires a point definition you can use any of the point defining commands, as you can with the other parameter definitions. /RECALL is a general use command to insert the last defined parameter of the type required by the syntax. The /USE command inserts the value of a variable of the appropriate type.

When there is a parameter referenced by the command syntax you can

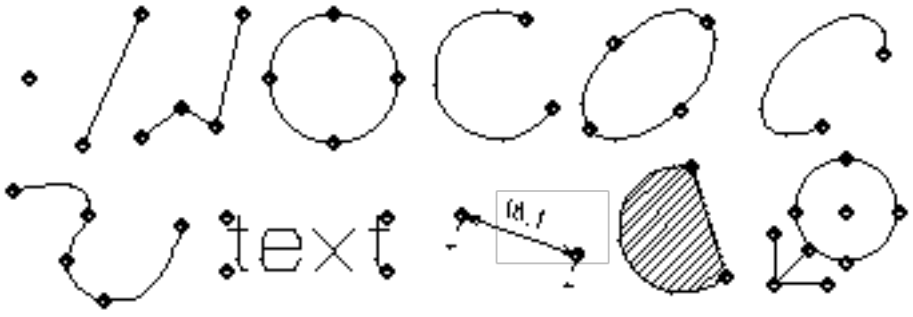
- ☐ enter a constant;
- ☐ use an expression;
- ☐ /USE a variable;
- ☐ /RECALL the last used value of that type;
- ☐ use parameter defining commands that follow.

Point Defining Commands

In the case of keyboard input topCAD uses the entered values as x and y coordinates. The /RECALL command defines the last used coordinate type value.

EXTREMUM single

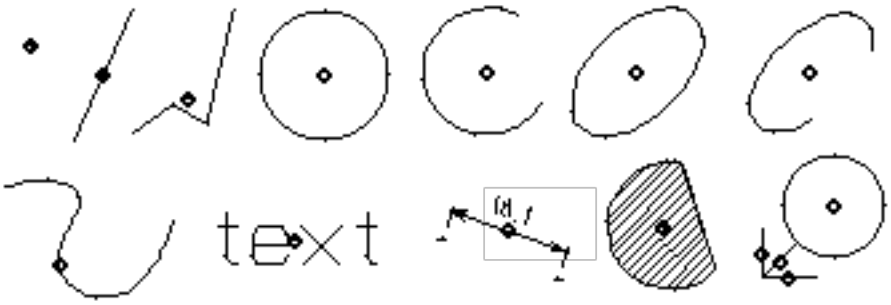
The nearest the pick endpoint of the object:



single: a point near the object.

CENTER single

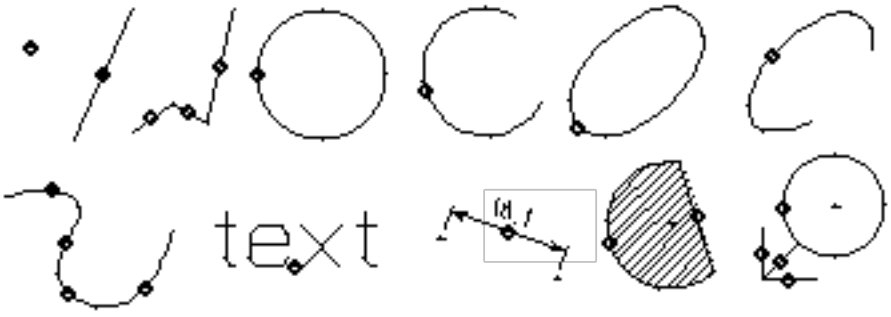
The nearest the pick center point of the object:



single: a point near the object.

MIDDLE single

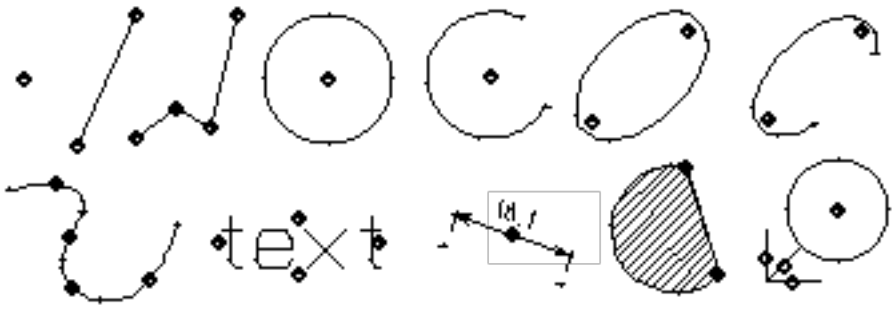
The nearest the pick midpoint of the object:



single: a point near the object.

FOCUS single

The nearest the focal point of the object:



single: a point near the object.

INTERSECTION single1 single2

The intersection point of the two objects closest to the given point. If there is no real intersection point between the two objects, topCAD tries to extend the objects.

single1: a point near one of the objects;
single2: a point near the other object.

MINTERSECTION single

The intersection point of the selected object closest to the given point.

single: a point near the object.

[X] length1
[Y] length2

Defines a point by absolute Cartesian (orthogonal) coordinates

length1: the x coordinate of the point;
length2: the y coordinate of the point.

POLAR length angle

Defines a point by absolute polar coordinates.

length: the length coordinate of the point;
angle: the angle coordinate of the point.

DX length1 DY length2

Defines a point by Cartesian (orthogonal) coordinates relative to the recently entered point.

length1: the relative x coordinate of the point;
length2: the relative y coordinate of the point.

DPOLAR length angle

Defines a point by polar coordinates relative to the recently entered point.

length: the relative length coordinate of the point;
angle: the relative angle coordinate of the point.

CXLATED length1 length2 coord

Defines a point by orthogonal coordinates relative to the given one.

length1: the relative x coordinate of the point;
length2: the relative y coordinate of the point;
coord: the reference point.

CXPOLAR length angle coord

Defines a point by polar coordinates relative to the given one.

length: the relative length coordinate of the point;
angle: the relative angle coordinate of the point;
coord: the reference point.

CXFORMED trans coord

Defines a point by transformation of the coordinates of the given one.

trans: the transformation defining the point;
coord: the reference point.

NEAREST single

The closest point of the object to the given point.

single: a point near the object.

CDIVIDED num single

Defines a point dividing the object according to the given proportion from the endpoint nearer the given point. The result will be the dividing point.

num: the dividing factor ($0 \leq num \leq 100$);
single: a point near the object.

CDISTANCE length single

Defines a point measuring the given distance along the object from the endpoint nearer the given point. The result will be the dividing point.

length: the distance of the dividing point from the endpoint nearer the picked point;
single: a point near the object.

MDISTANCE length single

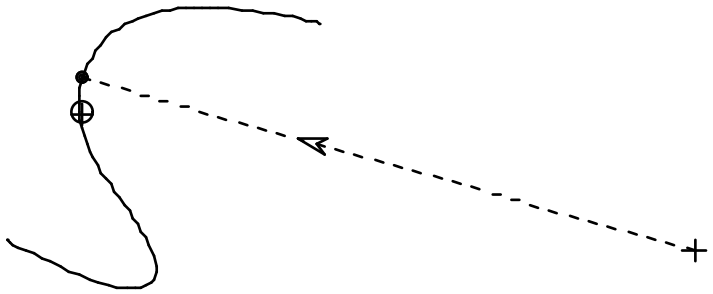
Defines a point measuring the given distance along the object from its midpoint. The result will be the dividing point.

length: the distance of the dividing point from the midpoint;
single: a point near the object.

TPERPENDICULAR coord single

Defines a point as the perpendicular projection of the selected point to the selected object nearest the selection point.

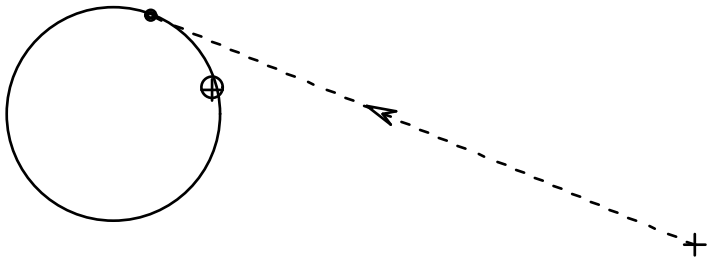
coord: defines the point to be projected;
single: a point near the object.



TTANGENT coord single

Defines a point as the tangential point on the line between the selected point and the selected object nearest the selection point.

coord: defines the point to be projected;
single: a point near the object.



ORIGO

Defines a point at the origin of the coordinate system.

BOTTOMLEFT

Defines a point at the bottom left corner of the drawing window.

TOPRIGHT

Defines a point at the top right corner of the drawing window.

CRTCENTER

Defines a point at the center of the drawing window.

USE var

Defines a point using the values stored in the given variable.

var: the name of the variable.

E.g.: /VCOORD corner1 100 100
/VCOORD corner2 200 200

You have defined two coordinate-type variables.

/ZOOM /IN /USE corner1 /USE corner2

The window to be ZOOMed IN is defined by the two coordinate variables corner1 and corner2.

RECALL

Uses the last defined point.

Length Defining Commands

In the case of keyboard input, topCAD uses the entered length. /RECALL defines the last used length type value.

PERIMETER single

Defines a length as the total length or the perimeter of the selected object:

- ☐ point: zero;
- ☐ line: length;
- ☐ polyline: the total length;
- ☐ circle: the perimeter;
- ☐ circular arc: the perimeter;
- ☐ ellipse: the perimeter;
- ☐ elliptic arc: the perimeter;
- ☐ spline: the total length;
- ☐ text: the perimeter of the enclosing rectangle;
- ☐ hatch: the perimeter of the boundary;
- ☐ dimension: the dimensioned length;
- ☐ symbol: the perimeter of the enclosing rectangle.

single: a point near the object.

SIZE single

Defines a length as the characteristic size of the selected object:

- ☐ point: zero;
- ☐ line: length;
- ☐ polyline: the length of the selected part;
- ☐ circle: radius;
- ☐ circular arc: radius;
- ☐ ellipse: half major axis;
- ☐ elliptic arc: half major axis;
- ☐ spline: the length of the selected part;
- ☐ text: character height;
- ☐ hatch: the offset (the distance between the lines);
- ☐ dimension: the dimensioned size;
- ☐ symbol: the width of the enclosing rectangle.

single: a point near the object.

LDISTANCE coord1 coord2

Defines a length as the distance between two given points.

coord1: one of the points;
coord2: the other point.

LSCALED num length

Defines a length as the given length multiplied by the given number.

num: the multiplication factor;
length: the length to be multiplied.

ACCUMULATED < length >

Defines a length as the sum of the given lengths.

length: a length to add.

REDUCED length1 < length2 >

Defines a length as the difference between the first length and the sum of all other lengths: length=length1-(length2+length3+ ...).

length1: the length given first;
length2: lengths to be summarized and subtracted from length1.

XC coord

Defines a length as the x coordinate of a point.

coord: the point whose x coordinate will be used.

YC coord

Defines a length as the y coordinate of a point.

coord: the point whose y coordinate will be used.

USE var

Defines a length using the value of the given variable. In the case of a length variable the command uses the value stored in the variable. In the case of a coordinate variable the variable is treated as a vector starting from the origin; the value used is the length of the vector.

var: the name of the variable.

RECALL

Uses the last defined length.

Angle Defining Commands

In the case of keyboard input, topCAD uses the entered angle. /RECALL defines the last used angular value.

APARALLEL single

Defines an angle as the angle of the selected object (or its tangent) at its nearest point to the given point.

single: a point near the object whose angle is to be used.

APERPENDICULAR single

Defines an angle adding 90° to the angle of the selected object (or its tangent) at its nearest point to the given point.

single: a point near the object whose angle is to be used.

AROTATED angle single

Defines an angle adding the defined angle to the angle of the selected object (or its tangent) at its nearest point to the given point.

angle: the rotation angle;
single: a point near the object whose angle is to be used.

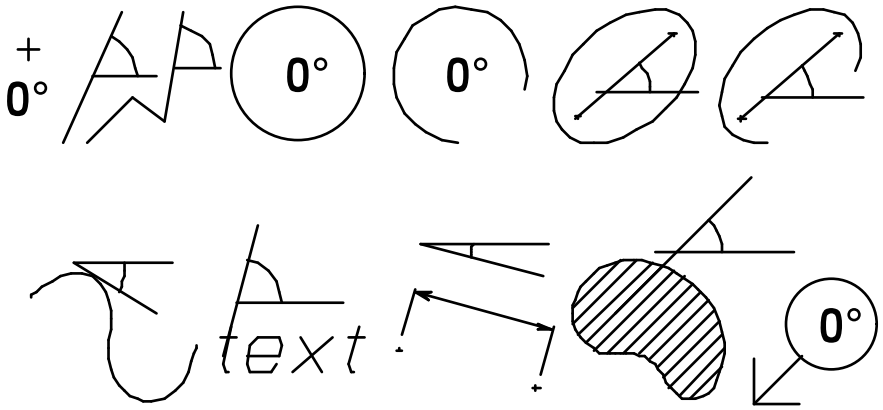
ADIVIDED num angle

Defines an angle dividing the specified angle by the given number.

num: the divider;
angle: the angle to be divided.

ASTRUCTURED single

The characteristic angle of the object:



single: a point near the object whose characteristic angle is to be used.

AINTERSECTION single1 single2

Defines an angle as the angle between the tangents of two intersecting objects at their intersection point closest to the given points. If there is no intersection point between the two objects, topCAD tries to extend the objects. The angle is in that fragment of the plane where the first point is located.

single1: a point near one of the intersecting objects;
single2: a point near the other object.

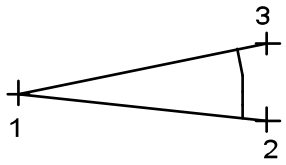
AMINTERSECTION single

Defines an angle as the angle between the tangents of the selected object and another one which intersects the first object closest to the given point. The angle is measured at the intersection point in that fragment of the plane where the first point is located.

single: a point near the object.

AGRAPHIC coord1 coord2 coord3

Defines an angle by the vertex and by a point in each boundary. For exact definition the point defining commands (/EXTREMUM, /NEAREST, /INTERSECTION etc.) must be used.



coord1: the vertex of the angle;
coord2: a point in one of the boundaries;
coord3: a point in the other boundary.

USE var

Defines an angle using the angle value stored in the given variable.

var: the name of the variable.

RECALL

Defines the last used angular value.

Text Defining Commands

These commands are for defining text as parameters of other commands. The text may exist in the drawing or come from the keyboard. Any text containing the space “ ”, or any other special characters must be entered between " " characters.

/RECALL defines the last used textual value.

CONCATENATED < text >

Defines a text joining the entered text strings.

text: a text string which will be a part of the concatenated text.

MULTILINE < text >

Defines a multiline text. Each entered text string will be a separate line in the multiline text.

text: a text string which will be a part of the multiline text.

TREPEAT num text

Defines a text repeating and joining the entered text *num* times.

num: the number of repetitions;
text: a text string to be multiplied.

SUBTEXT text num1 num2

Defines a text containing the part of the specified text from the *num1*st character up to the *num2*nd one.

text: the specified text;
num1: the sequence number of the first required character in the specified text;
num2: the sequence number of the last required character in the specified text.

TFILE file

Defines a text as the contents of a file.

file: the name of the file containing the text.

TDATE

Defines a text as the current date.

TTIME

Defines a text as the current time.

EXISTING single

Defines a text from the drawing.

single: a point near the text.

USE var

Defines a text using the text stored in the given variable.

var: the name of the variable.

RECALL

Defines the last used textual value.

Transformation Defining Commands

These commands are used to define the transformation parameter for the /AXFORM or the /LXFORM command. /RECALL defines the last used transformation.

XLATE coord1 coord2

Defines a shift transformation specifying the starting point and the endpoint of the shift vector.

coord1: the starting point of the vector;
coord2: the endpoint of the vector.

XLATE XY length1 length2

Defines a shift vector by its horizontal and vertical components.

length1: the horizontal offset;
length2: the vertical offset.

XLATE XPOLAR length angle

Defines a shift vector by its length and angle.

length: the length of the offset;
angle: the angle of the offset.

ROTATION angle coord

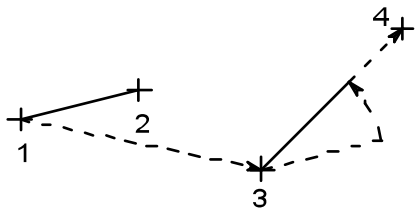
Defines a rotation around the given point by the *Preferred angle*.

angle: the rotation angle;
coord: the rotation center.

ROTOTRANS coord1 coord2 coord3 coord4

Defines a shift and a rotation. The shift vector starts from the first point and ends at the third point. The angle of rotation is the difference between the angles of the segments 3-4 and 1-2, the center of rotation is the first point.

coord1: a point to be moved (also serves as the rotation center);
coord2: the second point to be moved;
coord3: the endpoint of the shift vector, i.e., the new position for point 1;
coord4: a point which defines the angle of the segment 1-2 after the transformation.



MIRROR coord1 coord2

Defines a mirroring by the axis determined by the two points.

coord1: a point of the mirroring axis;
coord2: another point of the mirroring axis.

MIRROR ITEM single

Defines a mirroring by the given object. The object may be a point (rotation around this point by 180°), a line or a polyline (the part of the polyline closest to the pick).

single: a point near the mirroring object.

XSCALE num coord

Defines a scaling by the given factor and the given center.

num: the scaling factor;
coord: the center point of scaling.

XSCALE { X | Y } num coord

Defines a horizontal (or vertical) only scaling by the given factor and the given center.

X: horizontal only scaling;
Y: vertical only scaling;
num: the scaling factor;
coord: the center point of scaling.

LRATIO length1 length2 coord

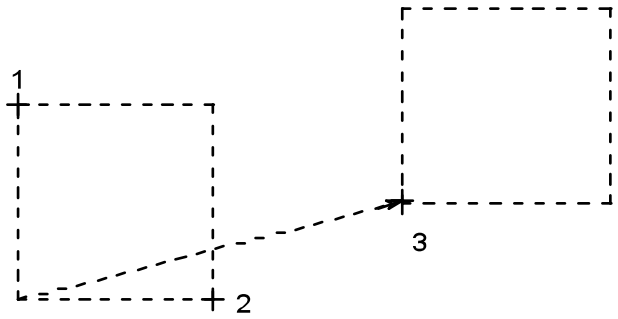
Defines a scaling by the given center with a scaling factor equal to the ratio *length1/length2*.

length1 and length2: numbers defining the scaling factor;
coord: the center point of scaling.

BOX XLATE coord1 { coord2 | BOXSIZES length1 length2 } { coord3 | XOFFSET length3 length4 }

Defines a shift as well as a window for the /MOVE /PART, /MOVE /CONTINUOUS, /STRETCH etc. commands. Instead of the coordinates it is possible to enter the absolute sizes of the box and the offset vector.

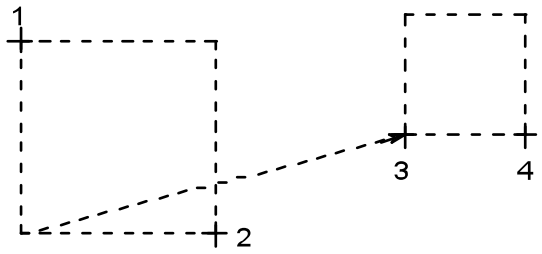
coord1: a corner of the window (the starting point of the movement vector);
coord2: the opposite corner of the original window; or
BOXSIZES: the window size will be given;
length1: the horizontal size of the window;
length2: the vertical size of the window;
coord3: the new position of the first corner of the window (the endpoint of the movement vector); or
XOFFSET: the movement will be defined by its horizontal and vertical components;
length3: the horizontal component of the movement vector;
length4: the vertical component of the movement vector.



BOX XSCALE coord1 { coord2 | BOXSIZES length1 length2 } { coord3 | XOFFSET length3 length4 } { coord4 | XFACTOR num }

Defines a shift and a scale as well as a window for the /MOVE PART, /MOVE CONTINUOUS, /STRETCH etc. commands. Instead of the coordinates it is possible to enter the absolute sizes of the box and the offset vector. It is also possible to enter the correct scale factor numerically.

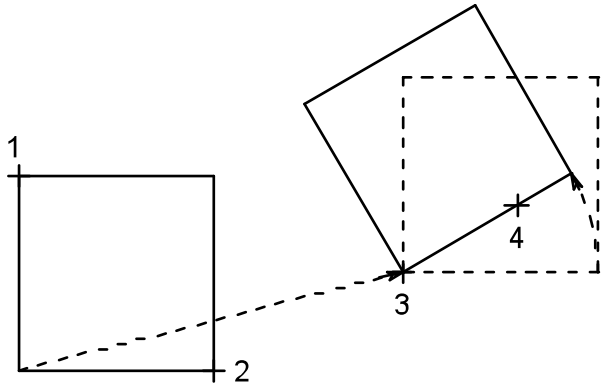
coord1: a corner of the original window (the starting point of the movement vector);
coord2: the opposite corner of the original window; or
BOXSIZES: the window size will be given;
length1: the horizontal size of the window;
length2: the vertical size of the window;
coord3: the new position of the first corner of the transformed window (the endpoint of the movement vector); or
XOFFSET: the movement will be defined by its horizontal and vertical components;
length3: the horizontal component of the movement vector;
length4: the vertical component of the movement vector;
coord4: the new position of the opposite corner of the transformed window; or
XFACTOR: the scaling will be defined numerically;
num: the scale factor.



BOX ROTATION coord1 { coord2 | BOXSIZES length1 length2 } { coord3 | XOFFSET length3 length4 } { coord4 | XANGLE angle }

Defines a shift and a rotation as well as a window for the /MOVE /PART, /MOVE /CONTINUOUS, /STRETCH etc. commands. Instead of the coordinates it is possible to enter the absolute sizes of the box and the offset vector. The correct angle value can also be entered.

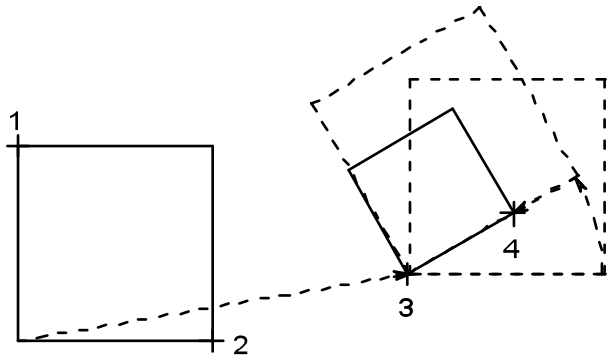
coord1: a corner of the original window (the starting point of the movement vector);
coord2: the opposite corner of the original window; or
BOXSIZES: the window size will be given;
length1: the horizontal size of the window;
length2: the vertical size of the window;
coord3: the new position of the first corner of the transformed window (the endpoint of the movement vector); or
XOFFSET: the movement will be defined by its horizontal and vertical components;
length3: the horizontal component of the movement vector;
length4: the vertical component of the movement vector;
coord4: the new position of the opposite corner of the transformed window to define the rotation graphically; or
XANGLE: the rotation will be defined numerically;
angle: the rotation angle.



BOX ROTOTRANS coord1 { coord2 | BOXSIZES length1 length2 } { coord3 | XOFFSET length3 length4 } { coord4 | XFACTOR num [XANGLE] angle | XANGLE angle [XFACTOR] num }

Defines a shift, a rotation and a scaling as well as a window for the /MOVE /PART, /MOVE /CONTINUOUS, /STRETCH etc. commands. Instead of the coordinates it is possible to enter the absolute sizes of the box and the offset vector. The correct angle value and the scale factor can also be entered numerically.

coord1: a corner of the original window (the starting point of the movement vector);
coord2: the opposite corner of the original window; or
BOXSIZES: the window size will be given;
length1: the horizontal size of the window;
length2: the vertical size of the window;
coord3: the new position of the first corner of the transformed window (the endpoint of the movement vector); or
XOFFSET: the movement will be defined by its horizontal and vertical components;
length3: the horizontal component of the movement vector;
length4: the vertical component of the movement vector;
coord4: the new position of the opposite corner of the transformed window to define the transformation graphically; or
XFACTOR: the scaling will be defined numerically;
num: the scale factor;
XANGLE: the rotation will be defined numerically;
angle: the rotation angle.



IDENT

Defines the coincidental transformation:

- ☐ no translation;
- ☐ no rotation;
- ☐ scale factors=1.

MULTITRANS < tran >

Combines the given transformations.

tran: definitions of different transformations.

MATRIX x11 x12 x13 x21 x22 x23

Defines a transformation matrix by elements.

USE var

Defines a transformation as the one stored in the given transformation type variable.

var: the name of the variable.

RECALL

Defines the last used transformation.

Selection Definitions

These object defining commands are used inside other commands, when an object type parameter is required. In the syntax, the object type parameter appears as single or group. A selection definition is used to define an object or a set of objects from the current drawing. The nesting command will, then, act upon those objects selected by the current selection definition. All of the selection defining commands but /XSELECT follow the command that define the action on the objects to be selected. The simplest selection definition is selecting an object by one of its points (simply clicking on it, or entering coordinate values).

Selection Defining Commands

XSELECT group

This command is for Macintosh-like selection, that is, when selection occurs first, then you define what to do with those objects selected. The command makes the objects appointed in the command line, or by clicking, selected. Then, a following command can act on the objects, for example, modify their color or line type.

group: a group of selected objects.

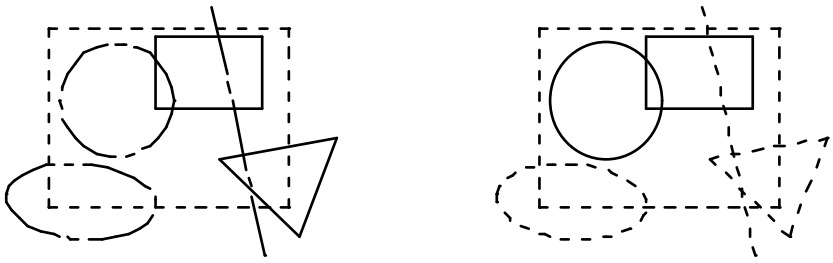
E.g.: /XSELECT /ALL /SCIRCLE /EXCEPT /SATTRIBUTES /SCOLOR 2 ; ;
/MOVE /SELECT ;

selects all circles of the drawing except for the red ones, and moves them (according to the preferred transformation).

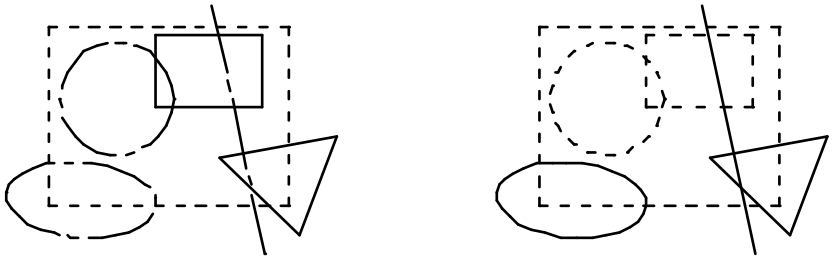
SWINDOW { IN | OUT | FRAME | INFRAME | OUTFRAME | NOFRAME } coord1 coord2

Selects objects by a selection window.

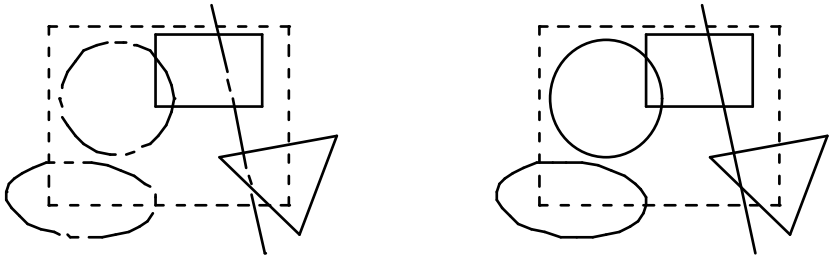
IN: selects all objects located entirely inside a window;
OUT: selects all objects located entirely outside a window;
FRAME: selects all objects which have a common point with the frame of a window;
INFRAME: selects all objects which have a common point with the frame of a window or locate entirely inside the window;
coord1: a corner of the window;
coord2: the opposite corner of the window.



/SWINDOW /IN



/SWINDOW /FRAME

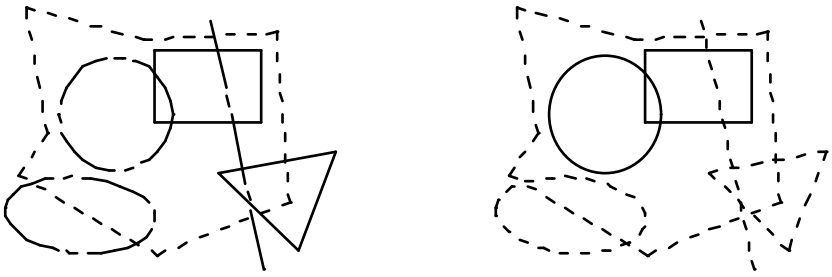


/SWINDOW /INFRAME

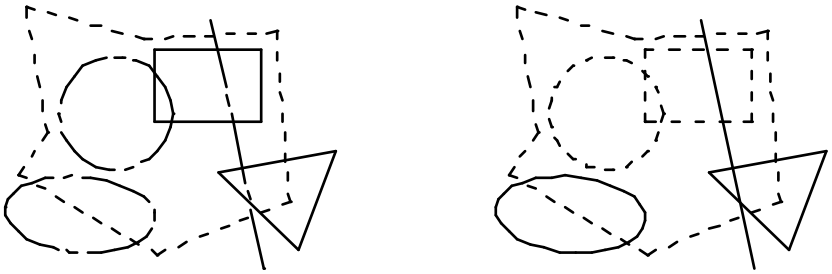
SPOLYGON { IN | OUT | FRAME | INFRAME | OUTFRAME | NOFRAME }
< coord >

Selects objects by an imaginary polygon.

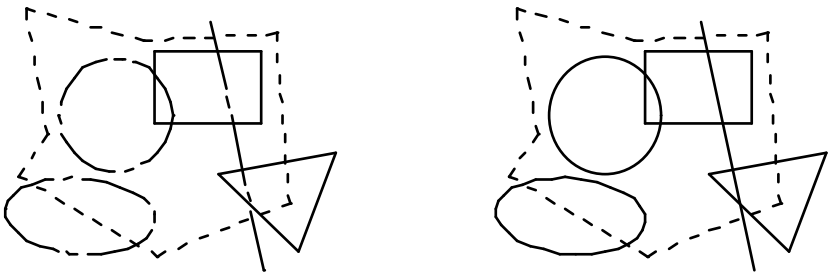
IN: selects all objects located entirely inside the polygon;
OUT: selects all objects located entirely outside the polygon;
FRAME: selects all objects which have a common point with the polygon;
INFRAME: selects all objects which have a common point with the polygon or locate entirely inside it;
coord: a corner of the window.



/SPOLYGON /IN



/SPOLYGON /FRAME



/SPOLYGON /INFRAME

CHAIN single

Selects all objects belonging to the selected chain. The chain must be closed. The way the objects are selected is similar to that in the /HATCH /CHAIN command.

single: a point near an object in the chain.

INTCHAIN single

Selects all objects belonging to the selected chain and its internal chains. The chains must be closed. The way the objects are selected is similar to that in the /HATCH /INTCHAIN command.

single: a point near an object in the chain.

OPENCHAIN single

Selects objects belonging to the selected open chain.

The whole chain or a part of the chain can be selected:

- pointing to the first or the last object in the chain nearer its free endpoint, the whole chain is selected;
- pointing to the first or the last object in the chain nearer its endpoint connected to the next object, only that object is selected;
- pointing to any other object in the chain, a part of the chain will be selected: from the chosen object to the end of the chain which is farther from the selected point than from the midpoint of the chosen segment.

single: a point near an object in the chain.

SNAME symb

Selects all positioned symbols with the given name.

symb: the name of the symbol to be selected.

SLAYER { ACTIVE | { GROUP | NGROUP } set }

Selects objects by layer.

ACTIVE: selects all the objects located on the active layer;
GROUP: selects all the objects located on one of the listed layers;
NGROUP: selects all the objects not located on any of the listed layers;
set: a set of layers.

SCOLOR { [GROUP] | NGROUP } set

Selects objects by color.

GROUP: selects all objects with one of the listed colors (by default);
NGROUP: selects all objects whose color is not on the list;
set: a set of logical color numbers.

SLTYPE { [GROUP] | NGROUP } set

Selects objects by line type.

GROUP: selects all objects with one of the listed line types (by default);
NGROUP: selects all objects whose line type is not on the list;
set: a set of logical line type numbers.

SLWIDTH [INTERVAL] { [GROUP] | NGROUP } set

Selects objects by line width.

INTERVAL: selects all objects whose line width is in one of the given line width ranges;
GROUP: selects all objects whose line width is on the list (by default);
NGROUP: selects all objects whose line width is not on the list;
set: a list of line width values, or pair of line width values as intervals (if /INTERVAL is included). When /INTERVAL is included in the command line, the interval should be given as follows: the smallest required line width, followed by the largest required line width.

EXCEPT group

Excludes the selected set of objects from the selection. The command acts as a logical NOT function.

group: selection of the objects to be excluded.

E.g.: /MOVE /SELECT /SWINDOW 280 150 110 30 /EXCEPT 220 130 ;

moves objects enclosed by the selection window, except for the one which passes through the point (220; 130).

AND group

Removes from the selection all objects not selected after the /AND. Only objects that fit the previous selection criteria *and* the one following /AND will be selected. The command acts as the logical AND operation.

group: selection of the objects.

E.g.: /MODIFY /LWIDTH 5 /ALL /SCIRCLE /AND /SWINDOW 0 0 350 475 ; ;

sets the line width of all circles inside the selection rectangle. The other circles in the drawing as well as any other type of object remain unchanged.

BEGIN < group >

Starts a new selection group. The consecutive /EXCEPT and /AND statements will not have any effect on the part of the selection entered before the /BEGIN.

group: selection of the objects.

E.g.: /MODIFY /COLOR 1
/SATTRIBUTES /SCOLOR 2 : AND
/BEGIN /ALL /SCIRCLE /ALL /SLINE ; ; ; ;

changes the color of all red circles and red lines to black.

Selection Definitions by Object Types

ALL { SPOINT | SLINE | SPOLYLINE | SCIRCLE | SCARC | SELLIPSE | SEARC | SSPLINE | STEXT | SDIMENSION | SHATCH | SSYMBOL | SITEM }

Selects all visible objects of the given type.

SPOINT, SLINE, ...: restricts the selection to all objects of the given object type (points, lines, and so on).

E.g.: /MOVE /SELECT /ALL /SCIRCLE ;

Moves all circles using the *Preferred Transformation*.

SPOINT < coord >

SLINE < coord >

SPOLYLINE < coord >

SCIRCLE < coord >

SCARC < coord >

SELLIPSE < coord >

SEARC < coord >

SSPLINE < coord >

STEXT < coord >

SDIMENSION < coord >

SHATCH < coord >

SSYMBOL < coord >

These commands restrict selection to one of the given type of objects. The selected object is the one nearest to the point within the tolerance circle.

coord: a point near the object.

SITEM coord

The command extends searching to all objects again after an erroneous restriction of type.

coord: a point near the object.

CHAPTER

4

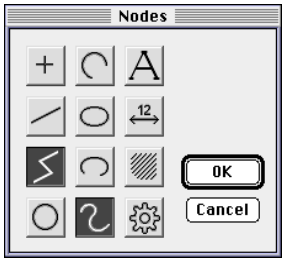
Macro/External Procedure Examples

Macro Examples

In the following exercises you can see some simple examples of the use of macros. All the macros and external procedures below make up a part of the topCAD Installation Kit and are copied onto disk at installation time. Two folders will be created, “Examples” and “Interface”, in the folder “Programmers Only” in the same folder as topCAD 2.0 itself.

Exercise 1

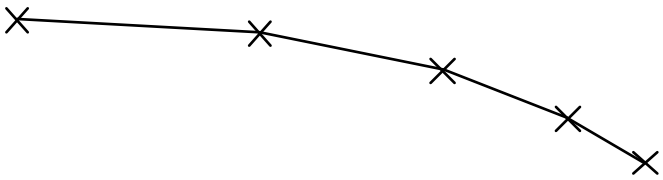
Let’s suppose you have to produce a gear with an NC machine tool. The tooth profile of the gear is represented by a spline in the drawing:



(Display of the nodes of the splines and polylines can be set by the *Nodes...* command in the View menu.)

The NC machine tool, however, cannot accept the spline, which therefore must be approximated by lines or arcs and the nodes of the resulting object must be stored numerically.

The spline can be converted into a polyline by the *Spline Make Polyline* command in the Modify menu:



As you can see, this is a poor approximation; the number of the polyline nodes should be increased.

This can be achieved, for example, by the following steps:

- ☐ divide the spline into, say, 30 equal parts,
- ☐ draw a polyline using the points dividing the spline as nodes of the polyline,
- ☐ delete the spline,
- ☐ redraw the drawing.

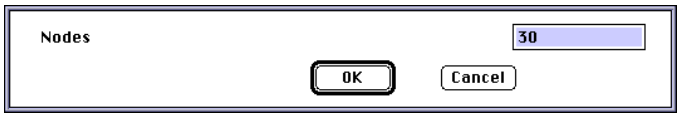
The macro below realizes these steps:

```
/dialog { /vint Nodes 30 } ;
/vcoord P 'Click spline to convert'
/vfloat rate 0
/polyline
/for i,0,(i<=Nodes)
      { rate = (i*100/Nodes)
        /cdivided (rate) (P.x) (P.y)
      } ; ;
/delete /select /sspline (P.x) (P.y) ; ;
/redraw
```

Let's inspect the commands in this macro line by line:

```
/dialog { /vint Nodes 30 } ;
```

Opens a dialog box showing the integer type variable Nodes and sets its default value to 30:



A new value for Nodes can be entered if necessary. Click on the OK button to keep the value at 30.

```
/vcoord P 'Click spline to convert'
```

Defines a coordinate type variable named P. The user defined prompt 'Click spline to convert' prompts to select the spline that should be converted into a polyline. The coordinate values assigned to P are defined by the click.

/vfloat rate 0

Defines the floating point type variable rate to store the current proportion of the two parts of the divided spline.

/polyline

Draws a polyline after defining the nodes and giving an ENTER. The nodes will be defined in the following commands. (This is why no ENTER (;) follows the keyword /POLYLINE.)

The next lines form a loop in order to divide the spline into 30 equal parts and define the 31 dividing points:

```
/for i,0,(i<=Nodes)
      { rate = (i*100/Nodes)
        /cdivided (rate) (P.x) (P.y)
      } ; ;
```

The loop starts with the /FOR loop keyword defining the loop and contains a series of commands between braces { } which are executed according to the condition in the /FOR statement.

/for i,0,(i<=Nodes)

Defines a loop. Contains a loop variable, its starting value and a condition.

The operation of the command:
the keyword /FOR instructs *topCAD* to analyze the condition in the parentheses. The command(s) between the braces are executed if the logical value of the condition is true. The loop variable is increased or decreased (in the current case it is increased by one) and the operation of the command starts again. If the logical value of the condition is false, *topCAD* skips the command(s) in braces and executes the command following the loop.

The elements of the command following the /FOR keyword:

i:	loop variable (integer);
0:	the starting value of the loop variable i
(i<=Nodes):	condition. If the value of i is less than or equal to the value of the variable Nodes, the commands standing after the /FOR statement between braces are executed;

{ rate = (i*100/Nodes)

The commands between the braces will be executed in all cases when the (i<=Nodes) condition is true.

The value of the proportion between the two parts of the spline divided by the current division point is assigned to the variable rate.

/cdivided (rate) (P.x) (P.y)

The /CDIVIDED command creates a point dividing the selected object with the ratio (rate = (i*100/Nodes)) from the endpoint nearer the selection (point P).

In our case the value of i starts from 0 and is increased by one until it reaches 30. The expression (i*100/Nodes) will take the following values expressed in percents:
0*100/30 defines the starting point of the spline,
1*100/30 defines the first dividing point at 1/30 of the spline,
2*100/30 defines the second dividing point at 2/30 of the spline, ...
(P.x) (P.y) the coordinates of the point P, select the spline to be divided.

The command, executed 31 times in the loop, defines 31 points on the spline dividing it into 30 parts of equal length. These points will be the nodes of the polyline approximating the spline.

} ;;

The "}" terminates the series of commands in the loop,
the first ";" (ENTER) completes the polyline, the second ";" (ENTER) completes the /POLYLINE command.

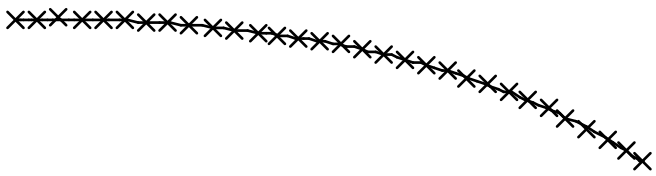
/delete /select /sspline (P.x) (P.y) ; ;

Deletes the spline nearby the point P. The /SELECT /SSPLINE command ensures that only the spline will be deleted (the polyline is also near the point P but it should not be deleted).
The first ";" (ENTER) terminates the selection, the second ";" (ENTER) completes the /DELETE command.

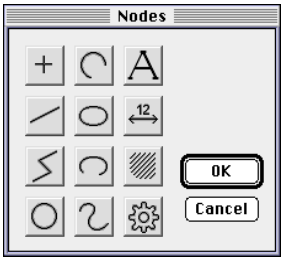
/redraw

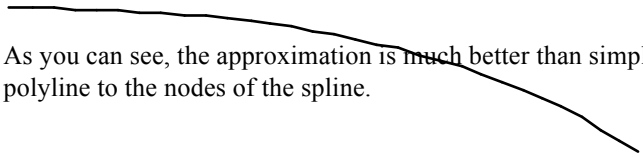
Freshens the screen redrawing its contents.

The execution of this macro gives this result:



Clear the nodes by the *Nodes* ...command in the View menu:

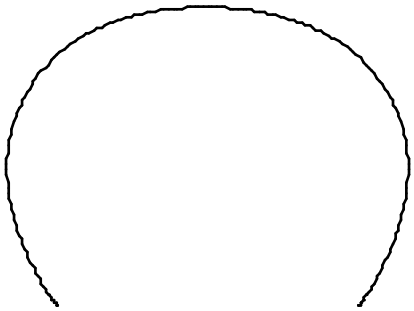




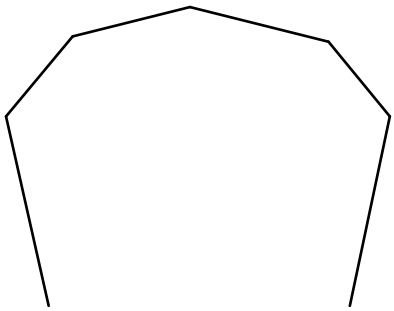
As you can see, the approximation is much better than simply fitting a polyline to the nodes of the spline.

Exercise 2

In the second example the curved part of the workpiece is:



Conversion of this spline into a polyline with the *Spline Make Polyline* command in the Modify menu results:



A much better approximation is needed again. The principle of the operation is similar to that in example 1, with slight differences:

- ☐ divide the spline into small equal parts,
- ☐ store the coordinates of the dividing points in two arrays,
- ☐ delete the spline,
- ☐ draw a smooth curve consisting of arcs, using the points that divide the spline.

Note: Because of the approximation with a smooth curve of arcs, when the coordinates of all the points dividing the spline are calculated and stored it is necessary to delete the spline before drawing the smooth curve. Otherwise, if the spline exists when the curve of arcs is being drawn, the point `P` used to select the spline may select the curve of arcs, causing confusion in the operation of the macro.

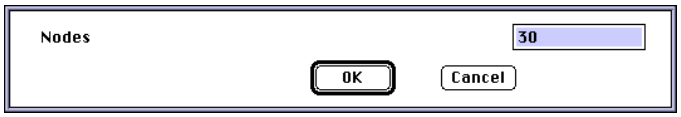
The macro which realizes this concept is:

```
/dialog { /vint Nodes 30; }
/vcoord P 'Click spline to convert'
/vcoord divpoint 0 0
/vfloat rate 0
/dimarr /vfloat Qx[Nodes+1] Qy[Nodes+1];
/for i,0,(i<=Nodes)
      { rate = (i*100/Nodes)
        /vcoord divpoint /cdivided (rate)
        (P.x) (P.y)
        Qx[i] = (divpoint.x)
        Qy[i] = (divpoint.y)
      } ;
/delete (P.x) (P.y) ;
/ccurve /smooth
/for i,0,(i<=Nodes)
      { (Qx[i]) (Qy[i]) } ; ;
```

The functions of the commands are:

/dialog { vint Nodes 30; }

Opens a dialog box with a default number of the dividing points:



Entering a value, you can change the precision of the approximation; a number less than 30 results in a poorer approximation and vice versa. Let's use this value; click on the OK button.

/vcoord P 'Click spline to convert'

Defines a coordinate type variable named P. The prompt 'Click spline to convert' prompts to select the spline to be converted into a smooth curve. The coordinate values assigned to P are defined by the click.

/vcoord divpoint 0 0

Defines the coordinate type variable divpoint storing the coordinates of the current dividing point and assigns to it the starting values 0,0.

```
/vfloat rate 0
```

Defines the floating point type variable rate storing the current proportion of the two parts of the divided spline.

```
/dimarr /vfloat Qx[Nodes+1] Qy[Nodes+1];
```

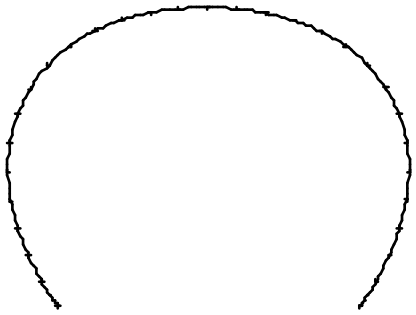
Defines two floating point type, one-dimension arrays named Qx for the x coordinates and Qy for the y coordinates of all dividing points. The size of the arrays is determined by the number of nodes (Nodes) of the approximating curve, which has been defined by the /DIALOG command.

Note that the size and the indexes of an array must be given between [ ] characters.

The next commands form a loop to calculate the coordinates of the 31 dividing points and store them in the arrays Qx and Qy:

```
/for i,0,(i<=Nodes)
      { rate = (i*100/Nodes)
   /vcoord divpoint /cdivided (rate)
                              (P.x) (P.y)
   Qx[i] = (divpoint.x)
   Qy[i] = (divpoint.y)
   };
```

The i^{th} elements of the floating point type arrays Qx and Qy will be assigned the coordinates of the i^{th} dividing point calculated by the /CDIVIDED command:



Note: this drawing does not appear when you execute the macro, it is just an explanatory figure.

The command uses the x and y coordinates ((P.x) (P.y)) of the point P to identify the spline to be divided.

The commands in the loop are:

/for i,0,(i<=Nodes)

The command line defines a loop with the loop variable i, its starting value (0) and the condition (i<=Nodes) .

The loop variable i will have the values 0,1,2, ..., 30.

{ rate = (i*100/Nodes)

The current value of the expression on the right side will be assigned to the variable named rate. This is the proportion of the two parts of the divided spline. The loop variable i will take the values 0,1,2, ..., 30.

/vcoord divpoint /cdivided (rate)
(P.x) (P.y)

Assigns the coordinates of the current division point of the object selected by the point P to the coordinate type variable divpoint.

Qx[i] = (divpoint.x)

Assigns the x coordinate of the current dividing point to the corresponding element of the floating point type array Qx.

Qy[i] = (divpoint.y)

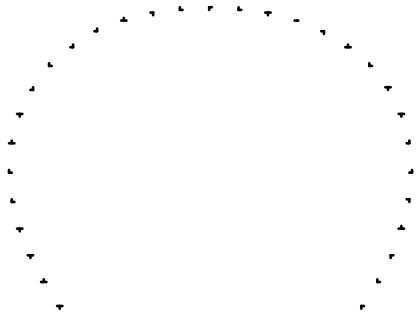
Assigns the y coordinate of the current dividing point to the corresponding element of the floating point type array Qy.

};

the "}" closes the loop, the ";" (ENTER) returns to the Command prompt.

/delete (P.x) (P.y);

Deletes the spline selected with the point P.
You don't need it any more since its dividing points are stored



Note: This drawing does not appear when you execute the macro, it is an explanatory figure only.

in the arrays Qx and Qy. This command also uses the point P to identify the object to be deleted.

/ccurve /smooth

Draws a smooth curve composed from arcs. It uses pairs of coordinates; the first three coordinate pairs define the first arc, each following coordinate pair defines the next arc.

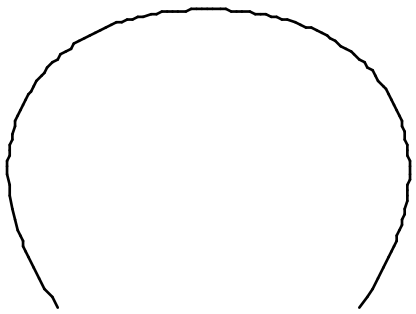
The coordinate pairs will be read from the arrays Qx and Qy.

Note that no ";" (ENTER) follows the command; consequently the following loop is embedded into it.

```
/for i,0,(i<=Nodes)
      { (Qx[i] ) (Qy[i] ) };;
```

The coordinate pairs of the points dividing the spline are read from the arrays Qx and Qy.

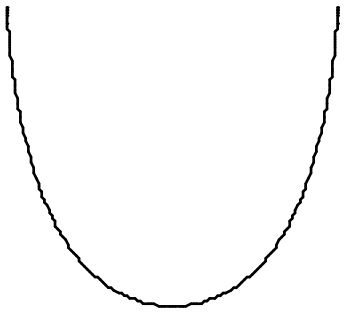
This loop reads the coordinate pairs from the arrays Qx and Qy. Remember that these are the coordinates of the 31 points dividing the spline. Now these points will be the parameters of the /CCURVE /SMOOTH command, i.e., the curve will be fitted to the dividing points of the spline:



The first ENTER (;) completes the curve, the second ENTER (;) quits the /CCURVE /SMOOTH command.

Exercise 3

In the third example the curved part of the workpiece looks like this:



In order to make a good approximation

- ☐ divide the spline into small equal parts,
- ☐ store the coordinates of the points dividing the spline in a file,
- ☐ draw a polyline using the division points of the spline as nodes for the polyline,
- ☐ delete the spline,
- ☐ redraw the drawing.

These operations can be performed with the following macro:

```
/dialog { /vint Nodes 30 ; }
/vcoord P 'Click spline to convert'
/openprint "::List:Spline Points"
/printf    "::List:Spline Points" "/polyline ";
/vfloat rate 0
/for i,0,(i<=Nodes)
 { rate = (i*100/Nodes)
   /vcoord Q /cdivide (rate) (P.x) (P.y)
   /printf  "::List:Spline Points"
            "%10.4f %10.4f" (Q.x) (Q.y) ;
 }
/printf    "::List:Spline Points" ";;";
/closefile "::List:Spline Points";
/delete /select /sspline (P.x) (P.y) ;
/call "::List:Spline Points"
/cancel
```

The functions of the commands are:

/dialog { /vint Nodes 30 ; }

Opens a dialog box with a default number of the dividing points:

Entering a greater value, you can improve the precision of the approximation; let's enter the value 40, then click on the OK button.

/vcoord P 'Click spline to convert'

Defines a coordinate type variable named P. The user defined prompt 'Click spline to convert' prompts to select the spline to be converted into a polyline. The x,y values assigned to P are defined by the click.

/openprint "::List:Spline Points"

Creates a file with the name "Spline Points" in the List folder and opens it for output. The coordinates of the points dividing the spline will be stored in this file.

/printf "::List:Spline Points" "/polyline ";

Loads the keyword /POLYLINE into the beginning of the file "Spline Points".

Later we will collect the coordinates of the points dividing the spline in this file; they will be used as the parameters of the /POLYLINE command. This file will be used as a macro when drawing the polyline approximating the spline.

/vfloat rate 0

Defines the floating point type variable rate which will store the proportion of the two parts of the spline divided by the current dividing point.

The following lines form a loop which defines the dividing points and puts their coordinates into the file “Spline Points”:

```
/for i,0,(i<=Nodes)
 { rate = (i*100/Nodes)
   /vcoord Q /cdivide (rate) (P.x) (P.y)
   /printf  "::List:Spline Points"
          "%10.4f %10.4f" (Q.x) (Q.y);
   }
```

/for i,0,(i<=Nodes)

Defines a loop with the loop variable i, its starting value (0) and the condition (i<=Nodes).

The loop variable i will take the values 0,1,2, ..., 40.

{ rate = (i*100/Nodes)

The current value of the expression on the right side will be assigned to the variable named rate. This is the proportion of the two parts of the divided spline. The loop variable i will take the values 0,1,2, ..., 40.

/vcoord Q /cdivide (rate) (P.x) (P.y)

Defines the i^{th} point dividing the spline and assigns its coordinates to the coordinate type variable Q. (P.x) and (P.y) are the x and y coordinates of the point P which identifies the object to be divided.

```
/printf  "::List:Spline Points"
       "%10.4f %10.4f" (Q.x) (Q.y);
```

Stores the coordinates of the point Q in the file called “Spline Points”. The command, according to the syntax rules of the programming language “C”, consists of a parameter list (%f: a floating point number) and a list with the same number of variables.

%10.4f: represents a 10-digit floating point number with 4 decimal digits.

} Closes the loop.

/printf "::List:Spline Points" " ; ;" ;

Stores two ENTERs at the end of the file “Spline Points”. One ENTER will complete the polyline approximating the spline, the other ENTER quits the /POLYLINE command.

/closefile "::List:Spline Points" ;

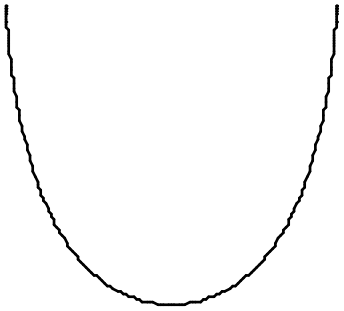
Closes the file “Spline Points”. This file, containing the keyword /POLYLINE followed by 41 pairs of coordinates and two ENTERs (;;), is a complete polyline definition command which can be used as a macro (see below).

/delete /select /sspline (P.x) (P.y);

Deletes the spline identified by the point P. The /SELECT /SSPLINE keywords are necessary if there are also other objects near the point P.

/call "::List:Spline Points"

The file “Spline Points” is called as a macro to draw the polyline approximating the spline, using the spline dividing points stored in the file as nodes:



As is shown here, you can call macros from any macro without any limitation.

/cancel

Returns to the Command prompt.

The content of the file “Spline Points” is a syntactically correct polyline definition command:

```
/polyline
   40.0000   200.0000
   40.3935   193.7700
   40.9313   187.5453
   41.5948   181.3402
   42.3887   175.1522
   43.3272   168.9752
   44.4283   162.8294
   45.7233   156.7218
   47.2560   150.6697
   49.0556   144.6916
   51.1483   138.8098
   53.5581   133.0509
   56.3132   127.4493
   59.4979   122.0810
   63.1693   117.0342
   67.3612   112.4117
   72.0751   108.3234
   77.2535   104.8426
   82.8544   102.0957
   88.8094   100.2429
   95.0000    99.5367
  101.1905   100.2429
  107.1456   102.0958
  112.7466   104.8427
  117.9249   108.3234
  122.6388   112.4117
  126.8307   117.0342
  130.5022   122.0812
  133.6868   127.4493
  136.4419   133.0509
  138.8517   138.8098
  140.9444   144.6916
  142.7440   150.6697
  144.2767   156.7218
  145.5717   162.8294
  146.6726   168.9738
  147.6084   175.1316
  148.4053   181.3412
  149.0701   187.5595
  149.6067   193.7725
  150.0000   200.0000
;;
```

Exercise 4

In mechanical engineering you must often draw nuts. Let‘s see how a tool for drawing any standard nut can be created in topCAD. A solution to this problem is to write some macros.

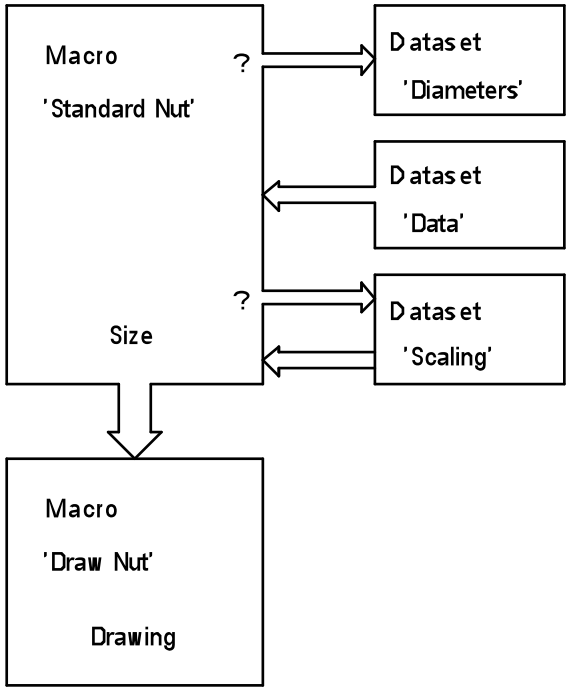
The functions of this nut drawing macro set:

- ☐ show the possible sizes according to the industrial standards, which enables you to choose the required one,
- ☐ create as a symbol the nut of the chosen size,
- ☐ locate the symbol.

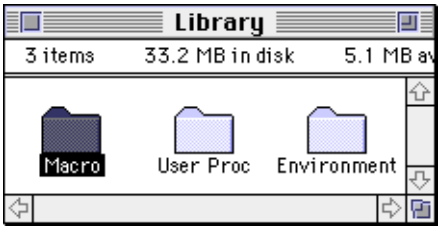
In order to draw a nut using this set of macros, only the following steps are required:

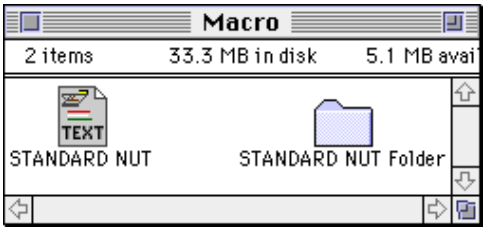
- ☐ definition of the size of the nut,
- ☐ definition of the scaling,
- ☐ locating the nut.

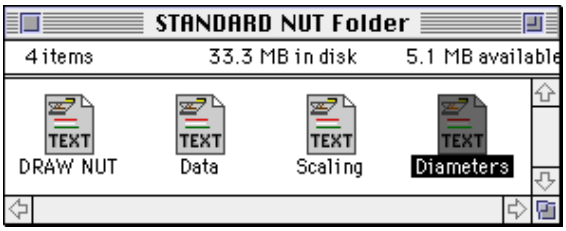
The structure of a macro system for the above task is:



The file structure is:







The macro “Standard Nut”:

```
#************************************************
#
#
#
# Standard Nut
#
#
#
#************************************************
#

#
# variable definitions for the dimensions of the
# nut
/vfloat da  0
/vfloat dwa 0
/vfloat ea  0
/vfloat pa  0
/vfloat mag 0
/vfloat m1a 0
/vfloat sa  0
/vfloat ga  0
/vinteger i 0

#
# open a dialog box to choose a diameter
#
/vinteger ind '#d:Standard Nut Folder:Diameters'

/vtext file ":Standard Nut Folder:Data"

#
# define the first part of symbol name
#
/vtext alap "Standard Nut M"
/vinteger eo (EOF(file))
/if (eo>=0)
    { /closefile /use file }

#
# open the data file
#
/openscan /use file
/vinteger eo (EOF(file))
/if (eo<0)
    { /exit }
```

```
#
# read the nut data from smallest until selected
#
/for i 0 (i<ind)
     { /scanf /use file "%f" da "%f" pa "%f" dwa
                        "%f" ea "%f" mag "%f" m1a
                        "%f" sa "%f" ga  "%l"
spec; }
/vinteger eo (EOF(file))
/if (eo>0)
    { /exit }
/closefile /use file

#
# create a symbol name for the nut
#
/vtext snev /concat /use alap /use spec;

#
# call the drawing macro
#
/call ":Standard Nut Folder:Draw Nut"
```

The macro “Draw Nut”:

```
#************************************************
#
#
#
# Draw Nut
#
#
#
#************************************************
#

#
# define the nut as a symbol
#
/symbdef /use snev
/vint e (err)
/vfloat mm 0
/cancel
/if (0=e)
    {
      /autosave /off
      /attributes /put /all 9
      /layer /active /only 255;

   #
   # calculate the dimensions of the nut
   #
      /vcoord  oa -1000 -1000
      /vlength c (sa/0.866)
      /vlength w (oa.x-mag+c/20)
      /vlength v (oa.x-mag)
      /vlength u (oa.x-c/20)
```

```
   #
   # draw nut
   #

   /line /single
      (w) (oa.y-c/2) (u) (oa.y-c/2)
      (w) (oa.y-c/4) (u) (oa.y-c/4)
      (w) (oa.y+c/4) (u) (oa.y+c/4)
      (w) (oa.y+c/2) (u) (oa.y+c/2)
      (oa.x) (oa.y-c*.375) (oa.x) (oa.y+c*.375)
      (v) (oa.y-c*.375) (v) (oa.y+c*.375);

   /carc /p3
      (w) (oa.y-c/2) (v) (oa.y-c*.375) (w)
      (oa.y-c/4)
      /recall (v) (oa.y) (w) (oa.y+c/4)
      /recall (v) (oa.y+c*.375) (w) (oa.y+c/2)
      (u) (oa.y-c/2) (oa.x) (oa.y-c*.375) (u)
(oa.y-c/4)
      /recall (oa.x) (oa.y) (u) (oa.y+c/4)
      /recall (oa.x) (oa.y+c*.375) (u) (oa.y+c/2);

/autosave /on

   #
   # define symbol
   #
   /symbdef /use snev
   /slayer /active;
      (oa.x) (oa.y);;

   #
   # delete nut
   #
   /delete  (oa.x)  (oa.y);
   /layer /active 1;
   /layer /off 255;
   /attributes /get /all 9
   }
/cancel

/autosave /off
```

```
#
# set scale
#
/global /vfloat ma (ma)
/if (ma=0)
    {
    /vinteger mam '#d:Standard Nut Folder:Scaling'
    /switch (mam)
      {
         /case 1   { ma = 100}
         /case 2   { ma = 50}
         /case 3   { ma = 10}
         /case 4   { ma = 5}
         /case 5   { ma = 2}
         /case 6   { ma = 1}
         /case 7   { ma = 0.5}
         /case 8   { ma = 0.2}
         /case 9   { ma = 0.1}
         /case 10  { ma = 0.02}
         /case 11  { ma = 0.01}
         /default  { vfloat ma 'Scale'}}
         }
mm = (1.0/ma)
/dscale (1.0/mm)
```

```
#
# position symbol snapped to the nearest
# intersection of the selected object
# and aligned to it
#
/position /symbol
   /use snev
   /while (1) {
      /vcoord co1 /origo
      /vangle a1 /aparallel
       'Position the nut or click Cancel'
      /vcoord co1 /recall
        /if (co1.x=0) { /break }
        /vcoord co2 /minters /recall
        /vlength dx (co2.x-co1.x)
        /vlength dy (co2.y-co1.y)
        /vangle  da (0)
        /if (dx=0) { dx = (0.01) }
        /if (dx<0) { da = (180) }
        /vangle a2 (atn(dy/dx)+da)
        /if (abs(a2-a1)>1) { a1 = (a1+180) }
        /axform /multitrans  /xscale (mm) /origo
                /rotation (a1) /origo;
        (co2.x) (co2.y)
        };;
/autosave /on
/cancel
```

The dataset “Diameters”:

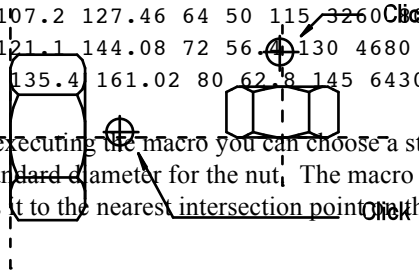
```
 M5
 M6
 M8
 M10
 M12
 M16
 M20
(M22)
 M24
(M27)
 M30
(M33)
 M36
 M39
 M42
(M45)
 M48
(M52)
 M56
(M60)
 M64
 M72x6
 M80x6
 M90x6
 M100x6
```

The dataset “Scaling”:

```
M 1:100
M 1:50
M 1:10
M 1:5
M 1:2
M 1:1
M 2:1
M 5:1
M 10:1
M 50:1
M 100:1
```

The dataset “Data”:

```
5 0.86 6.7 8.63 4 2.7 8 1.11 5
6 1 8.7 10.89 5 3.5 10 2.32 6
8 1.25 11.5 14.2 6.5 4.6 13 4.62 8
10 1.5 15.5 18.72 8 5.8 17 10.9 10
12 1.75 17.2 20.88 10 7.4 19 15.9 12
16 2 22 26.17 13 9.7 24 30.8 16
20 2.5 27.7 32.95 16 12.1 30 60.3 20
22 2.5 29.5 35.03 18 13.7 32 80.2 22
24 3 33.2 39.55 19 14.4 36 103 24
27 3 38 45.2 22 16.8 41 154 27
30 3.5 42.7 50.85 24 18.4 46 216 30
33 3.5 46.5 55.37 26 20 50 271 33
36 4 51.1 60.79 29 22.4 55 369 36
39 4 55.9 66.44 31 23.8 60 472 39
42 4.5 59.9 71.3 34 26.2 65 610 42
45 4.5 64.7 76.95 36 27.8 70 750 45
48 5 69.4 82.6 38 29.4 75 924 48
52 5 74.2 88.25 42 32.6 80 1130 52
56 5.5 78.7 93.56 45 35 85 1350 56
60 6 83.4 99.23 48 37.4 90 1600 60
64 6 88.2 104.86 51 39.6 95 1880 64
72 6 97.7 116.16 58 45.2 105 2520 72
80 6 107.2 127.46 64 50 115 3260 80
90 6 121.1 144.08 72 56.4 130 4680 90
100 6 135.4 161.02 80 62.8 145 6430 100
```



When executing the macro you can choose a standard diameter or give a non-standard diameter for the nut. The macro creates the nut as a symbol and fits it to the nearest intersection point on the selected object:

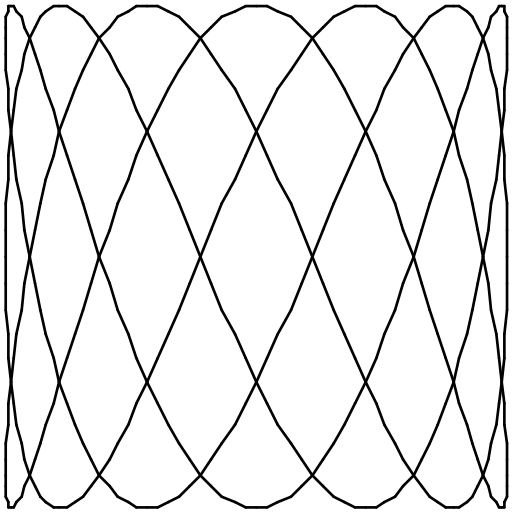
External Procedures

The following examples are mainly for those who are familiar with programming.

In topCAD you can use, apart from topCAD macros, external procedures written in the popular high-level programming languages:
MPW C, MPW Pascal, and Language Systems FORTRAN V1.2.1 under MPW 3.2 or later. This facility may help you to create fast-running procedures realizing complex functions such as iterations, etc. Let us see a simple (Exercise 5) and a little bit more complex (Exercise 6) example.

Exercise 5

The task is to draw the well-known Lissajous curve:



We have a C program, a Pascal program, a FORTRAN program and a topCAD macro, all performing the same task. You can compare the solutions (particularly the running times!).

The source programs:

- ☐ Lissajous.c
- ☐ Lissajous.p
- ☐ Lissajous.f

the command files:

- ☐ C&Link
- ☐ Pascal&Link
- ☐ FORTRAN&Link (see below)

the compiled and linked stand-alone applications:

- ☐ LissajousC
- ☐ LissajousP
- ☐ LissajousF

  which can be called by the /USERPROC command from a topCAD macro

and the macros:

- ☐ Lissajous
- ☐ LissajousM

are copied onto disk at installation time. Two folders are created for them, “topExamples” and “topInterface”, and can be found in the same place as topCAD 2.0 itself.

Starting the macro “Lissajous”, you define the parameters of the curve and choose which routine should be executed:

```
#*********************************************#
#                                             #
# Lissajous - topCAD macro choosing a routine #
#                                             #
#*********************************************#
/vint a '#PHorizontal ?,1,2,3,5,7,11,13,17,19,23,31'
/vint b '#PVertical ?,1,2,3,5,7,11,13,17,19,23,31'
/vint i '#pLanguage?,C,Pascal,FORTRAN,Macro'
/switch (i)
   {
   /case 1 { /userproc LissajousC }
   /case 2 { /userproc LissajousP }
   /case 3 { /userproc LissajousF }
   /case 4 { /call     LissajousM }
   };
/List "Number of new objects: %d" (i);
```

Choosing

- "C"
- "Pascal"
- "FORTRAN" or
- "Macro",

one of the following programs or the macro will be executed:

```
/************************************************
/
/*
*/
/* Lissajous.c - C routine to draw curve
*/
/*
*/
/************************************************
/
# include   <types.h>
# include   <math.h>
# include   <SANE.h>
# include   <stdlib.h>
# include   "::topInterface:topProc.h"

void Lissajous ()

{  int        i, n;
   extended   t, x1, y1, x2, y2, x, y, a, b;

   a = GetVariable ("A");
   b = GetVariable ("B");
   SetVariable ("topColor", 3);
   n = 20*a*b;
   x = a/pi();
   y = b/pi();
   for (x1 = 100.0, y1 = 150.0, i = 1 ; i <= n ; i++)
       {
          t = i / 10.0;
          x2 = 100 + 50 * sin (t/x);
          y2 = 100 + 50 * cos (t/y);
          SetLine (NewObj(), x1, y1, x2, y2);
          x1 = x2 ; y1 = y2;
       }
   SetVariable ("I", i-1);
}
```

```
(*************************************************
)
(*
*)
(* Lissajous.p - Pascal routine to draw curve
*)
(*
*)
(*************************************************
)
UNIT   myUnit;

INTERFACE

USES
   {$U MemTypes.p}               MemTypes,
   {$U QuickDraw.p}              QuickDraw,
   {$U OSIntf.p}                 OSIntf,
   {$U ToolIntf.p}               ToolIntf,
   {$U PackIntf.p}               PackIntf,
   {$U ::topInterface:topProc.p} User;

PROCEDURE   Lissajous;

IMPLEMENTATION

PROCEDURE   Lissajous;
   VAR   n, i, m: INTEGER;
         t, x1, y1, x2, y2, a, b: EXTENDED;
   BEGIN
      a := GetVariable ('A');
      b := GetVariable ('B');
      SetVariable ('topColor', 4.0);
      x1 := 100.0  ;  y1 := 150.0;
      m := round(20*a*b);
      FOR i := 1 TO m DO BEGIN
          t := i / 10.0;
          x2 := 100 + 50 * SIN (t/a*PI);
          y2 := 100 + 50 * COS (t/b*PI);
          n := NewObj;
          SetLine (n, x1, y1, x2, y2);
          x1 := x2 ; y1 := y2;
      END;
   SetVariable ('I', i-1.0);
   END;
```

```
END.
```

```
C***********************************************
)
C*
*)
C* Lissajous.f - FORTRAN routine to draw curve
*)
C*
*)
C***********************************************
)
       subroutine Lissajous

       INCLUDE    '::topInterface:topProc.f'
       integer*4  i,m, n
       extended   t, x1, y1, x2, y2, a, b
       CHARACTER*1 NULL
       string      aa, bb, ii, color

       NULL = 0
       aa   = 'A'//NULL
       bb   = 'B'//NULL
       ii   = 'I'//NULL
       color= 'topColor'//NULL
       CALL FGetVariable (%ref(aa), %ref(a))
       CALL FGetVariable (%ref(bb), %ref(b))
       CALL SetVariable  (%ref(color), 5.0)
       x1 = 100.0
       y1 = 150.0
       m = 20*a*b
       do 20 i=1,m
            t  = i / 10.0
            x2 = 100.0 + 50.0 * SIN(t/a*PI)
            y2 = 100.0 + 50.0 * COS(t/b*PI)
            CALL FNewObj(%ref(n))
            CALL SetLine (n, x1, y1, x2, y2)
            x1 = x2
            y1 = y2
20     continue
       CALL SetVariable (%ref(ii), i-1.0)
       return
       end
```

The C, Pascal and FORTRAN programs above use the following topCAD interface routines:

- **GetVariable**: accepts a topCAD variable
- **SetVariable**: assigns a value to a topCAD variable
- **NewObj**: defines a new topCAD object
- **SetLine**: defines a line type object by its endpoints.

The list of topCAD interface routines can be found in the C header file named topProc.h. The routines are delivered together with the topCAD installation kit. They are copied onto disk at installation time into the "Interface" folder. This folder resides in the same location as the topCAD program itself.

Note: In FORTRAN we use subroutines instead of functions, e.g. instead of the above NewObj function the FNewObj subroutine is used.

```
#************************************************#
#                                                #
# LissajousM - topCAD macro to draw curve        #
#                                                #
#************************************************#
/vfloat x (a/180)
/vfloat y (b/180)
/vint i 0
/vint n 20*a*b
/undo off
/line 100 150
/color 2
/for i,1,(i<=n)
  { (50*sin(0.1*i/x)+100) (50*cos(0.1*i/y)+100) }
/vint i (i-1)
/undo on
/cancel
```

In order to compile and link a program, run one of the following procedures:

```
#*********************************************************#
#                                                         #
# C&Link - command file to compile and link the           #
#          C routine "Lissajous.c"                        #
#                                                         #
#*********************************************************#
Directory "::User Proc"
C  -r  -b -mc68020 -mc68881 -o   Lissajous.c.o    Lissajous.c
Link   -o   ::Environment:LissajousC                      ∂
       -c      '????'                                     ∂
       -t      'data'                                     ∂
       -d                                                 ∂
       -sg   Lissajous                                    ∂
       -m    Lissajous                                    ∂
       -rt   USER=0                                       ∂
       Lissajous.c.o                                      ∂
       ::topInterface:topLib.o                            ∂
       "{CLibraries}"CLib881.o                            ∂
       "{CLibraries}"CSANELib881.o
Delete -y Lissajous.c.o

#*********************************************************#
#                                                         #
# Pascal&Link - command file to compile and link the      #
#               Pascal routine "Lissajous.p"              #
#                                                         #
#*********************************************************#
Directory "::User Proc"
Pascal  -mc68020  -mc68881  -o  Lissajous.p.o   Lissajous.p
Link  -o  ::Environment:LissajousP                        ∂
    -c    '????'                                          ∂
    -t    'data'                                          ∂
    -d                                                    ∂
    -sg  Lissajous                                        ∂
    -m   LISSAJOUS                                        ∂
    -rt  USER=0                                           ∂
    Lissajous.p.o                                         ∂
    ::topInterface:topLib.o                               ∂
    "{Libraries}"Interface.o                              ∂
    "{PLibraries}"PasLib.o
Delete -y Lissajous.p.o
```

```
#*********************************************************#
#                                                         #
# FORTRAN&Link - command file to compile and link the     #
#          FORTRAN routine "Lissajous.f"                  #
#                                                         #
#*********************************************************#
Directory "::User Proc"
FORTRAN -mc68020 -mc68881 -extended -case Lissajous.f
Link   -o   ::Environment:LissajousF                    ∂
       -c      '????'                                   ∂
       -t      'data'                                   ∂
       -d                                               ∂
       -sg   Lissajous                                  ∂
       -m    Lissajous                                  ∂
       -rt   USER=0                                     ∂
       Lissajous.f.o                                    ∂
       "{Libraries}"Interface.o                         ∂
       "{FLibraries}"FORTRANLib.o                       ∂
       ::topInterface:topLib.o
Delete -y Lissajous.f.o
```

Exercise 6

The last task again is the approximation of a spline with arcs. We will do it calling the external procedure “SplineToArc” written in “C” (see below) which results in the fastest and most qualified solution.

All the files below are in the “Examples” folder which is created at installation time. The folder can be found in the same place as the topCAD 2.0 program itself.

The calling macro “Call SplineToArc”:

```
#************************************************
#
#
#
# Call SplineToArcC - macro executing SplineToArc #
#
#
#************************************************
#

#************************************************
*
/vint nSpline 0
/vint nArcs 0
/dialog { /vfloat EPS 0.1 }
/undo /off
/userproc SplineToArcC
/list "Converted # %d spline  to # %d arc"
       (nSpline) (nArcs);
/undo /on
#************************************************
*
```

It defines some variables:

nSpline: the number of splines,
nArcs: the number of arcs,
EPS: the tolerance: the difference between the spline and the arcs will be less than or equal to this value (this is the precision of the approximation),

calls the external procedure “SplineToArcC”,
lists the number of converted splines and the number of arcs in the curves.

The SplineToArc.c procedure:

```
/*****************************************************************/
/*                                                               */
/*         SplineToArc.c DEMO PROGRAM to topCAUSERPROC           */
/*                                                               */
/*                  SMOOTHING SPLINES WITH ARCS                  */
/*                                                               */
/*                                                               */
/* Saturday, 27 May 1992 0:32:08                                 */
/* Written by Szabó Lóránt  Graphisoft                           */
/*                                                               */
/*****************************************************************/



/*****************************************************************/
/*                                                               */
/*                      INCLUDE HEADER FILES                     */
/*                                                               */
/*****************************************************************/

# include   <types.h>
# include   <math.h>
# include   <SANE.h>
# include   "::topInterface:topProc.h"
```

```
/*************************************************************/
/*                                                           */
/*                          DEFINITIONS                      */
/*                                                           */
/*************************************************************/

#define     EpsDeg    0.15
#define     TotDeg    360.0
#define     RadToDeg  (180.0/pi())
#define     PREC      20

void   SplineToArc    ();

int    Convert        (int Id);

COORD  Horner         (extended t,
                       COHandle a,
                       COHandle b,
                       COHandle c,
                       COHandle d);

int    CreateArc      (COORD     leftPoint,
                       COORD     middlePoint,
                       COORD     rightPoint,
                       COORD    *Origo,
                       extended *Radius,
                       extended *Alpha,
                       extended *Delta);

int    TestDistance   (COHandle  Coeff0,
                       COHandle  Coeff1,
                       COHandle  Coeff2,
                       COHandle  Coeff3,
                       extended  tBeg,
                       extended  tEnd,
                       COORD     Origo,
                       extended  Radius,
                       extended  Alpha,
                       extended  Delta,
                       extended  epsilon);
```

```
/*************************************************************/
/*                                                           */
/*                      MAIN ENTRY POINT                     */
/*                                                           */
/*************************************************************/

void SplineToArc ()

{ int nSpline, nArcs, next;

   SetVariable ("topColor",  1);    /* Set black color          */
   SetVariable ("topLTYPE",  1);    /* Set normal line type     */
   SetVariable ("topLWIDTH", 0);    /* Set narrowest line width */
   for (next = iFirst, nSpline = nArcs = 0; ; )
     {
       next = NextObj (next);       /* Look for the next object */
       if (next == iEnd) break;     /* If last then break       */
       if (ObjType (next) == kSpline)
           nArcs += Convert (next), nSpline++;
       }                            /* If spline call convert   */
   SetVariable ("nArcs",  nArcs);   /* Number of created arcs   */
   SetVariable ("nSpline", nSpline);/* Number of splines        */
}
```

```
/*************************************************************/
/*                                                           */
/*                  CONVERT A SPLINE TO ARCS                 */
/*                                                           */
/*************************************************************/

int  Convert (id)

int  id;

{
   COORD    origo,  Origo;        /* Previous and current origo */
   extended radius, Radius;       /* Previous and current radius*/
   extended alpha,  Alpha;        /* Previous and current angle */
   extended delta,  Delta;        /* Previous and current range */
   COHandle Coeff0, Coeff1, Coeff2, Coeff3;
                                  /* Spline coefficients */
   COORD    leftPoint;            /* Left point of arc          */
   COORD    middlePoint;          /* Middle point               */
   COORD    rightPoint;           /* Right point of arc         */
   int      Nodes;                /* Number of nodes on spline  */
   extended resol;                /* Spline resolution          */
   extended col;                  /* Color                      */
   extended T1, T2, t2;           /* Work variables for loop    */
   int      nArcs;                /* Number of created arcs     */
   int      first, ret;           /* Codes                      */
   extended Eps;                  /* Precision                  */

   nArcs = 0;
   col = GetVariable ("topColor");
   SetVariable ("topColor", ++col);
   Eps = GetVariable ("EPS");
   Eps = fabs (Eps);
   if (Eps > 0.5) Eps = 0.5;
   if (Eps < 0.0001) Eps = 0.0001;
   Coeff0 = (COHandle) NewHandle (0);
   Coeff1 = (COHandle) NewHandle (0);
   Coeff2 = (COHandle) NewHandle (0);
   Coeff3 = (COHandle) NewHandle (0);
   GetPSpline (id, &resol, &Nodes, Coeff0, Coeff1, Coeff2, Coeff3);
   leftPoint = **Coeff0;
   first = true;
```

```
  for (T1 = 0.0, T2 = Eps; T2 < Nodes - 1 + Eps/2.0; T2 += Eps)
     {
      if (T2 > Nodes - 1) T2 = Nodes - 1;
      middlePoint = Horner (T1 + (T2 - T1)/2.0,
                            Coeff3, Coeff2, Coeff1, Coeff0);
      rightPoint  = Horner (T2,Coeff3, Coeff2, Coeff1, Coeff0);
      ret = CreateArc (leftPoint, middlePoint, rightPoint,
                       &Origo, &Radius, &Alpha, &Delta);
      if (!ret)
          { SetLine (NewObj (), leftPoint.x,  leftPoint.y,
                                rightPoint.x, rightPoint.y);
            nArcs++;
            first = true;
            leftPoint = rightPoint;
            T1 = T2;
            continue; }
            ret = TestDistance (Coeff0, Coeff1, Coeff2, Coeff3,
                                T1, T2, Origo, Radius, Alpha,
                                Delta,  Eps);
      if ((ret = (ret && T2 != Nodes - 1)) || first)
           origo  = Origo,
           radius = Radius,
           alpha  = Alpha,
           delta  = Delta,
           t2     = T2;
      if (ret) continue;
      SetArc (NewObj (), origo.x, origo.y, radius, alpha, delta);
      nArcs++;
      first = true;
      leftPoint = rightPoint;
      T1 = T2 = t2;
      }
   DelObj (id);
   DisposHandle ((Handle) Coeff0);
   DisposHandle ((Handle) Coeff1);
   DisposHandle ((Handle) Coeff2);
   DisposHandle ((Handle) Coeff3);

   return  nArcs;
}
```

```
/*****************************************************************/
/*                                                               */
/*                      Mr.  HORNER's POLINOM                    */
/*                                                               */
/*****************************************************************/

COORD Horner (s, a, b, c, d)

COHandle a, b, c, d;
extended s;

{
   COORD    q;
   extended t;
   int      i;

   t = s - floor (s);
   i = s;
   if (t == 0.0 && i) i--, t = 1.0;
   q.x = (((*a+i)->x*t + (*b+i)->x)*t + (*c+i)->x)*t + (*d+i)->x;
   q.y = (((*a+i)->y*t + (*b+i)->y)*t + (*c+i)->y)*t + (*d+i)->y;
   return   q;
}
```

```
/*************************************************************/
/*                                                           */
/*                        CREATE AN ARC                      */
/*                                                           */
/*************************************************************/

int  CreateArc (leftPoint, middlePoint, rightPoint,
                Origo, Radius, Alpha, Delta)

   COORD          leftPoint;
   COORD          middlePoint;
   COORD          rightPoint;
   COORD         *Origo;
   extended      *Radius;
   extended      *Alpha;
   extended      *Delta;

{
   extended    dx1, dy1;
   extended    dx2, dy2;
   extended    v1, v2;
   extended    det;
   extended    d1, d2;
   extended    s;
   extended    phi, beta;
   Boolean     change;

   dx1 = middlePoint.x - leftPoint.x;
   dy1 = middlePoint.y - leftPoint.y;
   dx2 = rightPoint.x  - leftPoint.x;
   dy2 = rightPoint.y  - leftPoint.y;
   det = 4.0*(dx1*dy2  - dx2*dy1);
   if (fabs (det) < 0.000005) return false;
   v1 = dx1*(middlePoint.x + leftPoint.x) +
        dy1*(middlePoint.y + leftPoint.y);
   v2 = dx2*(rightPoint.x  + leftPoint.x) +
        dy2*(rightPoint.y  + leftPoint.y);
   d1 = 2.0*(v1*dy2 - v2*dy1);
   d2 = 2.0*(v2*dx1 - v1*dx2);
   Origo->x = d1/det;
   Origo->y = d2/det;
```

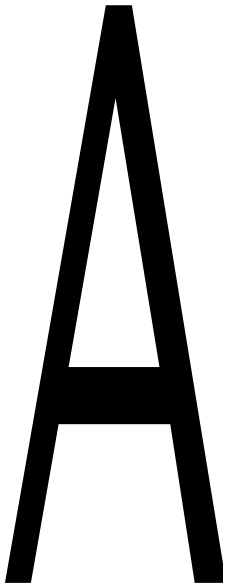
```
 *Alpha = atan2 (leftPoint.y   - Origo->y,
                 leftPoint.x   - Origo->x)*RadToDeg;
   phi    = atan2 (middlePoint.y - Origo->y,
                 middlePoint.x - Origo->x)*RadToDeg;
   beta   = atan2 (rightPoint.y  - Origo->y,
                 rightPoint.x  - Origo->x)*RadToDeg;
   *Radius= hypot (rightPoint.x  - Origo->x,
                 rightPoint.y  - Origo->y);
   if (beta < *Alpha) s = beta, beta = *Alpha, *Alpha = s;
   if (*Alpha < 0.0) *Alpha += TotDeg;
   if (phi < 0.0) phi += TotDeg;
   if (beta < 0.0) beta += TotDeg;
   if (fabs (*Alpha - beta) < EpsDeg)
       change = (*Alpha - phi <= EpsDeg && phi - beta <= EpsDeg
                 || *Alpha - phi - TotDeg <= EpsDeg &&
                    phi + TotDeg - beta <= EpsDeg
                 || *Alpha - phi + TotDeg <= EpsDeg &&
                    phi - TotDeg - beta <= EpsDeg
                 || beta - Alpha > EpsDeg ? false : true);
     else if (*Alpha > beta)
       change = (beta < phi && phi < *Alpha ||
                 beta < phi + TotDeg && phi + TotDeg < *Alpha
                 || beta < phi - TotDeg && phi - TotDeg <
                    *Alpha ? true : false);
     else
       change = (*Alpha <= phi && phi <= beta ||
                 *Alpha <= phi + TotDeg && phi + TotDeg
                 <= beta ||
                 *Alpha <= phi - TotDeg && phi - TotDeg
                 <= beta ? false : true);
   if (change) s = beta, beta = *Alpha,*Alpha = s - TotDeg;
   if (*Alpha < 0.0) *Alpha += TotDeg;
   *Delta = beta - *Alpha;
   if (*Delta < 0.0) *Delta += TotDeg;
   return true;
}
```

```
/*************************************************************/
/*                                                           */
/*       CALCULATE THE DISTANCE BETWEEN ARC AND SPLINE       */
/*                                                           */
/*************************************************************/

int TestDistance (Coeff0, Coeff1, Coeff2, Coeff3,
                  tBeg, tEnd, Origo, Radius, Alpha, Delta, Eps)

COHandle  Coeff0;
COHandle  Coeff1;
COHandle  Coeff2;
COHandle  Coeff3;
extended  tBeg;
extended  tEnd;
COORD     Origo;
extended  Radius;
extended  Alpha;
extended  Delta;
extended  Eps;

{
   int      i, n, ii, reverse;
   extended area, angle, t;
   COORD    points [2*PREC+1], arcb, arce, spline;

   n = 1.0/Eps;
   if (n > PREC) n = PREC;

   spline = Horner (tBeg, Coeff3, Coeff2, Coeff1, Coeff0);
   arcb.x = Origo.x + Radius*cos(Alpha/RadToDeg);
   arcb.y = Origo.y + Radius*sin(Alpha/RadToDeg);
   arce.x = Origo.x + Radius*cos((Alpha+Delta)/RadToDeg);
   arce.y = Origo.y + Radius*sin((Alpha+Delta)/RadToDeg);

   reverse = hypot (spline.x-arcb.x, spline.y-arcb.y) <
             hypot (spline.x-arce.x, spline.y-arce.y);
```

```
    for (i = 0; i <= n; i++)
         angle = (Alpha+i*Delta/n)/RadToDeg,
         points [i].x = Origo.x + Radius*cos(angle),
         points [i].y = Origo.y + Radius*sin(angle),
         t  = tBeg + i*(tEnd - tBeg)/n,
         ii = reverse ? 2*n - i : n + i,
         points [ii] = Horner (t, Coeff3, Coeff2, Coeff1, Coeff0);

    for (i = 1; i < n; i++)
         if (hypot (points[i].x-points[2*n-i].x,
             points[i].y-points[2*n-i].y) > 1.0)
         return false;

    for (i = 0, area = 0.0; i < 2*n; i++)
         ii = i == 2*n - 1 ? 0 : i + 1,
         area += (points[ii].x - points[i].x) *
                 (points[ii].y + points[i].y);

    t = Delta/RadToDeg*Radius;
    return fabs (area)/2.0/t < Eps;
}
```

APPENDIX

Keywords



This appendix lists the topCAD keywords in alphabetic order.

A0
A1
A2
A3
A4
A5
ABEGIN
ABORT
ACCUMULATED
ACTIVE
ADDNODE
ADDSYMB
ADIGITS
ADIVIDED
ADJUST
AEND
AGRAPHIC
AINTERSECTION
ALL
ALTERNATE
AMINTERSECTION
AND
ANGLE
ANTICYCLIC
APARALLEL
APERPENDICULAR
ARCHICAD
AROTATED
ARROW
ASCII
ASNAP
ASTRUCTURED
ATEXT
ATTRIBUTES
AUNIT
AUTONUMBER
AUTOSAVE
AUTOSCALE
AX
AXFORM
AXIS
AY

BACKWARD
BCARC
BCOLOR
BEARC
BEGIN
BEVEL
BITANGENT
BOTTOM
BOTTOMLEFT
BOUNDS
BOX
BOXSIZES
BREAK
BSEGMENT
BSPLINE

CALL
CANCEL
CANGLE
CARC
CASE
CATALOGUE
CAX
CAY
CCURVE
CDISTANCE
CDIVIDED
CENTER
CENTERED
CFONT
CGAP
CHAIN
CHEIGHT
CIRCLE
CLEAR
CLOSE
CLOSEFILE
CM
COLOR
COMPLEMENT
COMPOSITE
COMPRESS
CONCATENATE
CONNECT
CONSTLINE
CONTIGUOUS
CONTINUE
CONTINUOUS
COPY
CPOINT
CRTCENTER
CSLANT
CTANGENT
CUMULATIVE
CUT
CUTTING
CWIDTH
CXFORMED
CXLATED
CXPOLAR
CYCLIC

DAFORMAT
DANGLE
DATOLERANCE
DAUNIT
DEBUG
DECIMAL
DEFATTRIBUTES
DEFAULT
DEFHATCH
DEFINED
DELETE
DELNODE
DELPART
DETAILED
DEXTERNAL
DFIX
DFORMAT
DFPOINT
DFTANGENT
DIALOG
DIAMETER
DIGITS

DIMARRAY
DIMATTR
DIMENSION
DIMOPTION
DISTANCE
DISX
DISY
DMCOLOR
DMLWIDTH
DMS
DO
DOUBLE
DPARALLEL
DPOINT
DPOLAR
DRADIUS
DRAW
DSCALE
DTBOX
DTCOLOR
DTDIRECTION
DTLWIDTH
DTOLERANCE
DTPOSITION
DUMMY
DUPLICATE
DX
DXF
DY

EARC
EDIT
EFILE
EJECT
ELLIPSE
ELSE
ENDPOINT
ENTER
ENVIRONMENT
ERASE
ERROR
EXCEPT
EXISTING
EXIT
EXTREMUM

FDATE
FDISTANCE
FEET
FILLET
FINE
FIRST
FIXWIDTH
FOCUS
FOLDER
FOLLOWING
FONTDEF
FOR
FORMAL
FORWARD
FPOINT
FRAME
FSPECIAL
FTEXT
FTIME

GENATTR
GET
GLOBAL
GRID
GROUP

HATCH
HATCHCOMP
HATCHLINE
HATCHPATTERN
HATCHSYMBOL
HDIRECTION
HELP
HERE
HHATCH
HOFFSET
HOLE
HORIZONTAL
HOTSPOT
HPATTERN
HPGL
HSTEP
HV

ICURSOR
IDENT
IF
IGES
IN
INCH
INFRAME
INSCALE
INTCHAIN
INTERSECTION
INTERVAL
INTSHAPE
ITEM

JOURNAL
JUSTIFY

L
LAYER
LDISTANCE
LEFT
LENGTH
LGAP
LIKE
LINE
LIST
LISTFILE
LPERPENDICULAR
LRATIO
LSCALED
LTEXT
LTYPE
LWIDTH
LX
LXFORM
LY

M
MACRO
MAJOR
MATRIX
MDISTANCE
MERGE
MFONT
MIDDLE
MINOR
MINTERSECTION
MIRROR
MKIND
MM
MODIFY
MOVE
MOVENODE
MSIZE
MULTILINE
MULTITRANS
NAME

NC
NEAREST
NEW
NINTERVAL
NLENGTH
NO
NODES
NOFRAME
NORMAL
NSELECT

OFF
OFFSET
ON
ONLY
OPEN
OPENCHAIN
OPENED
OPENSCAN
ORIGINAL
ORIGO
OUT
OUTFRAME

P2TANGENT
P3
PAN
PARALLEL
PARS
PART
PASTE
PATHATCH
PAUSE
PBITANGENT
PDELETE
PDIMENSION
PEN
PERIMETER
PERPENDICULAR
PICT
PLOT
POINT
POLAR
POLYGON
POLYLINE
POSITION
POSTSCRIPT
PRECISION
PREVIOUS
PRINT
PRINTF
PRINTFILE
PRNOTES
PROGRESSIVE
PROJLINE
PROPORTIONAL
PRSUMMARY
PSCALE
PTANGENT
PURGE
PUT
PWIDTH
QSCALE
QUIT

RADIUS
RATIO
RECALL
RECTANGLE
REDRAW
REDUCED
REGULAR
RENAME
REPEAT
RESOLVE
RETURN
RGB
RIGHT
RINTERNAL
ROTATION
ROTOTRANS
ROUGH

SATTRIBUTES
SAVE
SCALE
SCANF
SCANFILE
SCARC
SCHAIN
SCIRCLE
SCOLOR
SDIMENSION
SEARC
SEARCHLIB
SECTION
SEGMENT
SELECT
SELLIPSE
SERIAL
SET
SETID
SFILE
SHAPE

SHATCH
SHATCHATTR
SHORTLIST
SHOW
SINGLE
SITEM
SIZE
SLAYER
SLICE
SLINE
SLTYPE
SLWIDTH
SMOOTH
SNAME
SNAP
SORIGIN
SPART
SPATTERN
SPLINE
SPOINT
SPOLYGON
SPOLYLINE
SPREADSHEET
SPRINT
SSITEM
SSPLINE
SSYMBOL
STACK
STEP
STEXT
STRAIGHT
STRETCH
SUBTEXT
SWINDOW
SWITCH
SYMBDEF
SYMBHATCH
SYMBNPAR
SYMBOL
SYMBPAR
SYMBTEXT

TABLET
TANGENT
TDATE
TDIRECTION
TEXT
TEXTATTR
TFILE
TIME
TLAYER
TMIRROR
TOLERANCE
TOP
TOPRIGHT
TORIGIN
TOTAL
TPERPENDICULAR
TREPEAT
TRITANGENT
TTANGENT
TTIME
TXREADABLE

ULINETYPE
UNDO
UNIT
UNLINK
USE
USEID
USERPROC

VANGLE
VCOORDINATE
VECTOR
VERTICAL
VFLOAT
VINTEGER
VISIBLE
VLENGTH
VTEXT
VXFORM

WAIT
WHILE
WINDOW
WITHPARAMETER

X
XANGLE
XC
XCENTER
XFACTOR
XLATE
XOFFSET
XPOLAR
XSCALE
XSELECT
XY

Y
YC
YES

ZOOM

?ACTUAL
?ANGLE
?ATTRIBUTES
?COORDINATE
?DISTANCE
?GRID
?HATCH
?ITEM
?LAYER
?LENGTH
?MENU
?SNAP
?SPACE
?SYMBOL
?TOLERANCE
?UNDO
?VARTAB

A P P E N D I X

B

Command Syntax

This appendix lists the syntax of topCAD commands in alphabetic order.

```
ABEGIN  angle
ABORT
ACCUMULATED  < length >
ADDSYMB  { ENTER | symb }  { ENTER | file }
ADIGITS  { LIKE  single | num }
ADIVIDED  num  angle
ADJUST  < single1  single2 >
ADJUST  FIRST  < single1  single2 >
ADJUST  GROUP  < single  group >
AEND  angle
AGRAPHIC  coord1  coord2  coord3
AINTERSECTION  single1  single2
ALL  { SPOINT | SLINE | SPOLYLINE | SCIRCLE | SCARC | SELLIPSE | SEARC | SSPLINE | STEXT |
        SDIMENSION | SHATCH | SSYMBOL | SITEM }
AMINTERSECTION  single
AND  group
ANGLE  angle
APARALLEL  single
APERPENDICULAR  single
AROTATED  angle  single
ASNAP  { < angle > | HV | OFF | ON | ALTERNATE }
ASTRUCTURED  single
ATEXT  text
ATTRIBUTES  DEFAULT  { ALL | ENTER | GENATTR | TEXTATTR | DIMATTR }
ATTRIBUTES  { PUT | GET }  { ALL | ENTER | GENATTR | TEXTATTR | DIMATTR }  { STACK | num
}
AUNIT  { DECIMAL | DMS }
AUTONUMBER
        { coord1  coord2 | ENTER }  { num1  num2 }
        { { LEFT | RIGHT }  { BOTTOM | TOP | ENTER } |
        { BOTTOM | TOP }  { LEFT | RIGHT | ENTER } | ENTER }
        { num3 | ENTER }  {num4 | ENTER }
        { { LEFT | RIGHT }  { BOTTOM | TOP | ENTER } |
        { BOTTOM | TOP }  { LEFT | RIGHT | ENTER } | ENTER }
        { num5 | ENTER }  { num6 | ENTER }  num7  group
AUTOSAVE  { TIME  num | STEP  num | OFF | ON | ALTERNATE }
AUTOSCALE
AXFORM  { ON | OFF | XCENTER | tran }

BCOLOR  num
BEGIN  < group >
BEVEL  < length  < single1  single2 > >
```

```
BOTTOMLEFT
BOX ROTATION coord1
        { coord2 | BOXSIZES length1 length2 }
        { coord3 | XOFFSET length3 length4 }
        { coord4 | XANGLE angle }
BOX ROTOTRANS coord1
        { coord2 | BOXSIZES length1 length2 }
        { coord3 | XOFFSET length3 length4 }
        { coord4 | XFACTOR num [XANGLE] angle | XANGLE angle [XFACTOR] num }
BOX XLATE coord1
        { coord2 | BOXSIZES length1 length2 }
        { coord3 | XOFFSET length3 length4 }
BOX XSCALE coord1
        { coord2 | BOXSIZES length1 length2 }
        { coord3 | XOFFSET length3 length4 }
        { coord4 | XFACTOR num }
BREAK

CALL file [ { < num | text > } ]
CANCEL
CARC < coord >
CARC BITANGENT < single1 single2 >
CARC CPOINT < coord1 coord2 coord3 >
CARC P3 < coord1 coord2 coord3 >
CATALOGUE ENTER
CATALOGUE SYMBOL [ DETAILED | ALL ] { symb | ENTER }
CATALOGUE { SFILE | DRAW | ENVIRONMENT | MACRO } { ENTER | file }
CCURVE CONTIGUOUS < coord1 coord2 coord3 < coord4 coord5 > >
CCURVE [SMOOTH] < coord1 coord2 coord3 < coord4 > >
CDISTANCE length single
CDIVIDED num single
CENTER single
CFONT { LIKE single | num }
CGAP { LIKE single | num }
CHEIGHT { LIKE single | length }
CIRCLE < coord >
CIRCLE BITANGENT < single1 single2 >
CIRCLE CPOINT < coord1 coord2 >
CIRCLE CTANGENT < coord single >
CIRCLE P2TANGENT < coord1 coord2 single >
CIRCLE P3 < coord1 coord2 coord3 >
CIRCLE PBITANGENT < coord single1 single2 >
```

CIRCLE TRITANGENT < single1 single2 single3 >
CLEAR group
CLOSE { YES | NO }
CLOSEFILE file
COLOR { LIKE single | num }
COMPLEMENT < single >
COMPOSITE name < num1 num2 num3 num4 num5 num6 >
COMPRESS { YES | NO }
CONCATENATE < text >
CONSTLINE < coord1 coord2 >
CONSTLINE BITANGENT < single1 single2 >
CONSTLINE DFPOINT < coord >
CONSTLINE DFTANGENT < single >
CONSTLINE DPARALLEL < single < length > >
CONSTLINE PARALLEL < single < coord > >
CONSTLINE PERPENDICULAR < single < coord > >
CONSTLINE PTANGENT < coord < single > >
CONSTLINE TANGENT < single >
CONTINUE
COPY { POSTSCRIPT } group
CRTCENTER
CSLANT { LIKE single | angle }
CUT { POSTSCRIPT } group
CUTTING < single1 single2 >
CUTTING FIRST < single1 single2 >
CUTTING SECTION < coord1 coord2 < coord2 > >
CUTTING { ALL | HERE } < single >
CWIDTH { LIKE single | num | PROPORTIONAL | FIXWIDTH }
CXFORMED trans coord
CXLATED length1 length2 coord
CXPOLAR length angle coord

DAFORMAT { LIKE single | text }
DATOLERANCE { ENTER | { LIKE single1 | text1 } { ENTER | LIKE single2 | text2 } }
DEFATTRIBUTES
{ SSPLINE | SPOLYLINE | STEXT | SSYMBOL | SCIRCLE | SCARC | SELLIPSE | SEARC |
SHATCH | SLINE | SDIMENSION | SPOINT } num
DELETE < single >
DELETE ALL
DELETE GROUP < group >
DELETE PART < coord1 coord2 >
DFORMAT { LIKE single | text }

DIALOG
{ < { /VINT I /VFLOAT I /VCOORD I /VANGLE I /VLENGTH I /VTEXT } ['text'] var value > }
DIGITS { LIKE single | num }
DIMARRAY
{ VINTEGER | VFLOAT | VLENGTH | VANGLE | VCOORDINATE | VXFORM | VTEXT } < var >
DIMENSION ARROW < coord1 coord2 { ENTER | coord3 { ENTER | coord4 } } >
DIMENSION DEXTERNAL < single coord >
DIMENSION DFIX < single coord >
DIMENSION DIAMETER < single coord >
DIMENSION DRADIUS < single coord >
DIMENSION PARALLEL < single1 single2 coord >
DIMENSION RINTERNAL < single coord >
DIMENSION [{ L | LX | LY }] < single coord >_
DIMENSION { AX | AY | CAX | CAY } < single coord >
DIMENSION { DANGLE | CANGLE } < single1 single2 coord >
DIMENSION { DISTANCE | DISX | DISY } < coord1 coord2 coord3 >
DIMENSION { DISTANCE | DISX | DISY } { SERIAL | CUMULATIVE | PARALLEL | PROGRESSIVE | SPART } < coord1 coord2 coord3 < coord4 > >
DIMOPTION num1 num2 num3 num4 num5
DMCOLOR { LIKE single | num }
DMLWIDTH { LIKE single | num }
DO { statements } num
DPOINT < single >
DPOLAR length angle
DSCALE { LIKE single | num }
DTBOX { LIKE single | num }
DTCOLOR { LIKE single | num }
DTDIRECTION { LIKE single | num }
DTLWIDTH { LIKE single | length }
DTOLERANCE { ENTER | { LIKE single1 | text1 } { ENTER | LIKE single2 | text2 } }
DTPOSITION { LIKE single | num }
DUMMY
DUPLICATE CONTINUOUS [coord1 coord2]
DUPLICATE GROUP group
DUPLICATE PART [coord1 coord2]
DUPLICATE single
DX length1
DY length2

```
EARC  < coord >
EARC  FPOINT  < coord1  coord2  coord3  coord4 >
EDIT  EFILE  file
EDIT  single
EJECT
ELLIPSE  < coord >
ELLIPSE  FDISTANCE  < coord1  coord2  length >
ELLIPSE  FPOINT  < coord1  coord2  coord3 >
ENTER
ERASE  SYMBOL  { ENTER | symb }  { YES | NO }
ERROR  { ON  I  OFF }
EXCEPT  group
EXISTING  single
EXIT
EXTREMUM  single

FDATE  num
FILLET  < single1  single2 >
FOCUS  single
FOLDER  { SFILE | DRAW | ENVIRONMENT | MACRO | PLOT | LIST }  text
FOLLOWING  num
FONTDEF  num  < char  [BOUNDS  num1  num2  num3  num4]  group >
FOR  [ var  num1  [ num2  [ num3 ] ] ]  { statements }
FSPECIAL  { ON | OFF }
        { ALL | ENTER |
        < { EXTREMUM | CENTER | MIDDLE | FOCUS | INTERSECTION | NEAREST } > }
        { ENTER | SSYMBOL | SHATCH }  ENTER
FTEXT  { ON | OFF }
FTIME  num

GLOBAL  attribute_setting
GRID  { ON | OFF | num1  num2  num3  num4  num5  length1  length2 }

HDIRECTION  { LIKE  single | angle }
HELP  file
HHATCH
        [ HATCHLINE | HATCHPATTERN | HATCHSYMBOL|  HATCHCOMP ]  BOUNDS
        < [ BSEGMENT  coord1  coord2 | BCARC  coord  length  angle1  angle2 |
        BEARC  coord  length1  length2  angle1  angle2  angle3 |
        BSPLINE  num1  num2...  num10 ] >
```

```
HHATCH
        { HATCHLINE | HATCHPATTERN | HATCHSYMBOL | HATCHCOMP }
        [ CHAIN | INTCHAIN | SHAPE | INTSHAPE ]  { IN | OUT }  [ num | name ]
        < group1  [ HOLE  group2 ] >
HOFFSET  { LIKE  single | length }
HPATTERN  { LIKE  single | num }
HSTEP  { LIKE  single | length }

ICURSOR  { ON | OFF | ALTERNATE }
IDENT
IF  num  { statements1 }  [ ELSE  { statements2 }]
INSCALE  num
INTCHAIN  single
INTERSECTION  single1  single2

JOURNAL  { ON | OFF | ALTERNATE }
JUSTIFY  { LIKE  single | LEFT | CENTERED | RIGHT }

LAYER  NAME  < num  name >
LAYER  { ACTIVE | ON | VISIBLE | OFF }  [ONLY]  set
LDISTANCE  coord1  coord2
LGAP  { LIKE  single | num }
LIKE  single
LINE  < < coord > >
LINE  AXIS  < coord1  < coord2  coord3 > >
LINE  AXIS  HV  < coord1  < coord2  coord3 > >
LINE  BITANGENT  < single1  single2 >
LINE  DPARALLEL  < single  < length > >
LINE  HORIZONTAL  < coord1  coord2 >
LINE  HV  < < coord > >
LINE  LPERPENDICULAR  < single  < coord > >
LINE  PARALLEL  < single  < coord > >
LINE  PERPENDICULAR  < coord  single >
LINE  PTANGENT  < coord  < single > >
LINE  SINGLE  < coord1  coord2 >
LINE  SINGLE  HV  < coord1  coord2 >
LINE  VERTICAL  < coord1  coord2 >
LIST  < text1  < { num | text2 } > >
LISTFILE  { ON | OFF }
LRATIO  length1  length2  coord
LSCALED  num  length
LTYPE  { LIKE  single | num }
```

```
LWIDTH  { LIKE  single | num | FINE | NORMAL }
LXFORM  { ON | OFF | tran }

MAJOR  length
MATRIX  x11  x12  x13  x21  x22  x23
MDISTANCE  length  single
MERGE  { SFILE | DRAW }  { ENTER | file }
MFONT  { LIKE  single | num }
MIDDLE  single
MINOR  length
MINTERSECTION  single
MIRROR  coord1  coord2
MIRROR  ITEM  single
MKIND  { LIKE  single | num }
MLAYER  layer  < group >
MODIFY  attribute  setting  < group >
MODIFY  CONNECT  << single > >
MODIFY  DAUNIT  { DECIMAL | DMS }  < group >
MODIFY  DIMENSION  < single  coord >
MODIFY  DIMOPTION  setting  < group >
MODIFY  ENDPOINT  < single  coord >
MODIFY  GROUP  < < attribute  setting >  group >
MODIFY  LENGTH  < single  coord >
MODIFY  LTEXT  text  < single >
MODIFY  NLENGTH  < single  length >
MODIFY  PDIMENSION  < single  coord >
MODIFY  POLYLINE  SLICE  < single >
MODIFY  POLYLINE  SMOOTH  < single >
MODIFY  SPLINE  ROUGH  < single >
MODIFY  SPLINE
        { { OPENED | CYCLIC | ANTICYCLIC }  < single>
        |  TANGENT  < single  { coord | VECTOR  length  angle } > }
MODIFY  SYMBNPAR  < single  num1  num2 >
MODIFY  SYMBPAR  < single  num  text >
MODIFY  TEXT  < single >
MODIFY  TOTAL  < single  < coord > >
MODIFY  { POLYLINE | SPLINE }  ADDNODE  < single  coord >
MODIFY  { POLYLINE | SPLINE }  DELNODE  < single >
MODIFY  { POLYLINE | SPLINE }  DELPART  < single >
MODIFY  { POLYLINE | SPLINE }  MOVENODE  < single  coord >
MOVE  CONTINUOUS  [coord1  coord2]
MOVE  GROUP  group
```

```
MOVE PART [coord1 coord2]
MOVE single
MSIZE { LIKE single | length }
MULTILINE < text >
MULTITRANS < tran >

NC < length < group > ENTER >
NC CHAIN < length < single > ENTER >
NC INTCHAIN < length < single > ENTER >
NC OPENCHAIN < length < single > ENTER >
NEAREST single
NEW { YES | NO }
NODES
        { ON | OFF | ALTERNATE }
        { ALL | ENTER |
        < { SSPLINE | SPOLYLINE | STEXT | SSYMBOL | SCIRCLE | SCARC | SELLIPSE | SEARC |
        SHATCH | SLINE | SDIMENSION } > ENTER }
NSELECT group
OFFSET < length < group > ENTER >
OFFSET CHAIN < length < single > ENTER >
OFFSET INTCHAIN < length < single > ENTER >
OFFSET OPENCHAIN < length < single > ENTER >
OPEN { SFILE | DRAW | DXF | IGES | PICT | HPGL | ENVIRONMENT } { file | ENTER }
OPENCHAIN single
ORIGO

PAN coord1 coord2
PARS < { VINTEGER | VFLOAT | VLENGTH | VANGLE | VCOORDINATE | VXFORM | VTEXT } <
        var > ENTER > ENTER >
PASTE group
PAUSE
PDELETE < single >
PEN num { PWIDTH length | set }
PERIMETER single
PLOT { A0 | A1 | A2 | A3 | A4 | A5 | length1 length2 } { file | .AOUT | .BOUT } [scale] [X offset Y
        offset] [YES | NO]
POINT < coord >
POLAR length angle
POLYGON REGULAR [ { IN | OUT } ] [ DOUBLE length ] < num < coord > >
POLYGON [ DOUBLE length ] < < coord > >
POLYLINE [ DOUBLE length ] < < coord > >
POSITION DRAW file < coord >
```

POSITION SYMBOL [HOTSPOT num]symb < coord >
PRECISION { LIKE single | num }
PREVIOUS num
PRINT scale { YES| NO } { YES| NO } [X offset] [Y offset]
PRINTF file < text1 < { num | text2 } > >
PRINTFILE file
PRNOTES text
PROJLINE { LIKE single | length | NO }
PRSUMMARY < num text >
PSCALE [SET num] coord
PURGE { YES | NO }

QSCALE { LWIDTH | LTYPE } { ON | OFF }
QUIT { YES | NO }

RADIUS length
RATIO { num | ENTER }
RECALL
RECALL
RECALL
RECALL
RECALL
RECTANGLE P3 [DOUBLE length] < coord1 coord2 coord3 >
RECTANGLE [DOUBLE length] < coord1 coord2 >
REDRAW
REDUCED length1 < length2 >
RENAME SYMBOL symb1 symb2
REPEAT num
RESOLVE { ENTER | file }
RETURN num
RGB { DEFAULT | < num red green blue > }
ROTATION angle coord
ROTOTRANS coord1 coord2 coord3 coord4

SAVE { SFILE | DRAW| DXF | IGES | PICT | ARCHICAD | HPGL | POSTSCRIPT | ASCII |
ENVIRONMENT } { file | ENTER }
SCALE num
SCANF file < text1 < var > >
SCANFILE file
SCARC < coord >
SCHAIN single
SCIRCLE < coord >

```
SCOLOR  { [GROUP] | NGROUP }  set
SDIMENSION  < coord >
SEARC  < coord >
SEARCHLIB  < file >
SELECT  group
SELLIPSE  < coord >
SETID  num
SHATCH  < coord >
SHOW  group
SITEM  coord
SIZE  single
SLAYER  { ACTIVE | { GROUP | NGROUP }  set }
SLINE  < coord >
SLTYPE  { [GROUP] | NGROUP }  set
SLWIDTH  [ INTERVAL | NINTERVAL ]  { [ GROUP ] | NGROUP }  set
SNAME  symb
SNAP  { length1  { ENTER | length2  { ENTER | length3  { ENTER | length4 } } } |
        ON | OFF | ALTERNATE | GRID  num }
SORIGIN  coord
SPATTERN  num
SPLINE  ANTICYCLIC  < < coord > >
SPLINE  CYCLIC  < < coord > >
SPLINE  TANGENT  < length1  angle1  length2  angle2  < coord > >
SPLINE  [OPENED]  < < coord > >
SPOINT  < coord >
SPOLYGON  { IN | OUT | FRAME | INFRAME | NOFRAME | OUTFRAME }  < coord >
SPOLYLINE  < coord >
SPRINT  var  < text1  < num  I  text2 >>
SSPLINE  < coord >
SSYMBOL  < coord >
STEXT  < coord >
STRETCH  [coord1  coord2]
SUBTEXT  text  num1  num2
SWINDOW  { IN | OUT | FRAME | INFRAME | NOFRAME | OUTFRAME }  coord1  coord2
SWITCH  num1  { < CASE  num2  [num3]  { statements1 } >  [DEFAULT  { statements2 }] }
SYMBDEF  HOTSPOT  name  group  < coord >
SYMBDEF  name  group  coord
SYMBNPAR  num1  num2
SYMBNPAR  { ON | FORMAL | OFF }
SYMBPAR  num  text
SYMBPAR  { ON | FORMAL | OFF }
SYMBTEXT  text
```

TABLET { ON| OFF}
TDATE
TDIRECTION { LIKE single | angle }
TEXT < coord >
TFILE file
TLAYER { ACTIVE | ORIGINAL }
TMIRROR { LIKE single | ON | OFF }
TOLERANCE num
TOPRIGHT
TORIGIN { LIKE single | num }
TPERPENDICULAR coord single
TREPEAT num text
TTIME
TXREADABLE { ON | OFF }

ULINETYPE num1 < num >
UNDO BACKWARD num
UNDO FORWARD num
UNDO { ON | OFF }
UNIT { MM | CM | M | { INCH | FEET } num }
UNLINK < single >
UNLINK GROUP < group >
USE var
USE var
USE var
USE var
USE var
USEID
USERPROC file

VANGLE var angle
VCOORDINATE var coord
VFLOAT var num
VINTEGER var num
VLENGTH var length
VTEXT var text
VXFORM var tran

WAIT sec
WHILE num { statements }
WINDOW coord1 coord2

XC coord
XLATE coord1 coord2
XLATE XPOLAR length angle
XLATE XY length1 length2
XSCALE num coord
XSCALE { X | Y } num coord
XSELECT group

YC coord

ZOOM IN coord1 coord2
ZOOM OUT coord1 coord2
[X] length1
[Y] length2

?ACTUAL
?ANGLE < angle >
?ATTRIBUTES
?COORDINATE < coord >
?DISTANCE < coord1 coord2 >
?GRID
?HATCH < single >
?ITEM [SPREADSHEET file]
?LAYER
?LENGTH < length >
?MENU
?SNAP
?SPACE set
?SYMBOL [SPREADSHEET file] { TOP | DETAILED }
{ DEFINED | [SHORTLIST | WITHPARAMETER] group }
?TOLERANCE
?UNDO
?VARTAB

A P P E N D I X

topCAD Variables

C

This appendix lists the topCAD variables in alphabetic order.

topCAD variables

topADIGITS	The number of decimals in the angular dimension tolerance
topALAYER	The number of the *Active Layer*
topALPHA	The *Preferred Start Angle*
topANGLE	The *Preferred Angle*
topASNAP	0: if *Angle Snap* is switched off 1: if *Angle Snap* is switched on with HV snap switched off 2: if *Angle Snap* is switched on with HV snap switched on
topAXISA	The *Preferred Major Half Axis*
topAXISB	The *Preferred Minor Half Axis*
topBCOLOR	The sequence number of *Background Color*
topBETA	The *Preferred End Angle*
topCFONT	The current font ID, according to the CFONT command
topCGAP	The *Character Gap* text attribute
topCHEIGHT	The *Character Height* text attribute
topCOLOR	The *Color* general attribute
topCSLANT	The *Character Slant* text attribute
topCURSOR	0: if the *Intelligent Cursor* is switched off, 1: if the *Intelligent Cursor* is switched on
topCWIDTH	The *Character Width* text attribute
topDIGITS	The number of decimals in linear dimension tolerance
topDISPLAY [2]	The size of screen in pixels
topDSCALE	The *Scaling Factor* dimension attribute
topDTBOX	0: if the *Boxed Text* dimension attribute is switched off 1: if the *Boxed Text* dimension attribute is switched on
topDTCOLOR	*Dimension Text Color*
topDTDIRECTION	*Dimension Text Direction:* is 0: standard 1: horizontal 2: vertical 3: parallel
topDTLWIDTH	*Dimension Text Line Width*

topDTPOSITION	*Dimension Text Position*
topHDIRECTION	The *Angle of Hatching*
topHOFFSET	The distance between two neighboring hatch lines
topHPATTERN	Serial number of the current *Hatch Pattern*
topHSTEP	Distance between hatch symbols being in line
topINSCALE	The *Input Scale* factor
topJOURNAL	The number of *Journal* files created during the current session
topJUSTIFY	Text attribute
topLAYER [1024]	The vector of layer statuses
topLEVEL	The current nesting level of a macro being executed
topLGAP	Text attribute: the distance between neighboring text lines
topLIST	0: if the list is directed to the screen 1: if the list is directed into a file
topLTYPE	The *Line Type* general attribute
topLWIDTH	The *Line Width* general attribute
topMFONT	The current font ID, according to the MFONT command
topMKIND	The current *Marker Type*
topMSIZE	The current *Marker Size*
topNOBJ	The *number of objects* in the current drawing
topNSYMB	The *number of symbols* in the current drawing
topNUMCOL	The *number of colors* of the monitor set by Monitors on the control panel
topPRECISION	The precision of splines
topPSCALE	The *Display Scale* factor
topRADIUS	The *Preferred Radius*
topRATIO	The current *Aspect Ratio*
topREPEAT	The *Preferred Repeat Factor*
topSCREEN [4]	Coordinates of the bottom-left and top-right corners of the current drawing window
topSNAP	0: if *Grid Snapping* is switched off, 1: if *Grid Snapping* is switched on
topSYMBPAR [2]	The values of the two numerical symbol parameters
topTDIRECTION	*Direction* text attribute
topTOLERANCE	Radius of the *Tolerance Circle*

topTORIGIN	*Text Origin* numbered bottom-up and left to right
topTRANS [3][3]	The *Preferred Transformation* matrix
topTXTREAD	0: text is also transformed no matter its orientation 1: text remains the right way up after a transformation
topULINETYPE [122][8]	128 custom line type definitions
topUNIT	The serial number of the current *Drawing Unit*
topVERSION	The *Version Number* of topCAD
topWINDOW [4]	Coordinates of the current window in the drawing in mms

APPENDIX

Functions

D

This appendix lists the topCAD functions in alphabetic order.

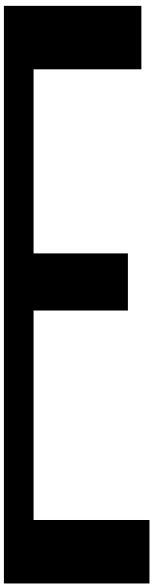
Functions

ABS	(expression)	The absolute value
INT	(expression)	The integer part
FRA	(expression)	The fractional part
SGN	(expression)	The sign function (return value is +1, 0, or -1)
SQR	(expression)	The square root function
ACS	(expression)	The arc cosine function (result in degrees)
ASN	(expression)	The arc sine function (result in degrees)
ATN	(expression)	The arc tangent function (result in degrees)
COS	(expression)	The cosine function (argument in degrees)
SIN	(expression)	The sine function (argument in degrees)
TAN	(expression)	The tangent function (argument in degrees)
EXP	(expression)	The exponential function
LGT	(expression)	The ten-based logarithm
LOG	(expression)	The natural logarithm
NOT	(expression)	The logical NOT function; returns: 0: if the argument ≠ 0 1: if the argument = 0
MAX	(expression <, expression >)	The maximal value function
MIN	(expression <, expression >)	The minimal value function
LEN	(text_variable)	The length of the text stored in the variable
POS	(text_variable, text_variable)	The position of the text stored in the second variable in the text stored in the first variable. The first index is 1. If the second text is not present in the first one, the result is zero.
CMP	(expression1,expression2)	Compares the value of expressions
OK		Returns: 0: if normal data input happened 1: after OK, CR or ENTER 2: after ABORT 3: after CANCEL In dialog, returns : 0: if normal input happened 3: after CANCEL
PI		The Ludolph's constant

REC		The latest entered numerical value (recall function)
RET		The return code of the previously called procedure
ERR		Non-zero if the last executed function failed, zero otherwise
EOF	(filename)	End of file; returns: -1: if the file is closed 0: if the file is opened 1: at end of file
RND		Produces a random number between (-1; 1)
SEL		The number of objects being selected.
VAR	(var)	The code of the variable type; returns: 0: if the variable does not exist 1: if integer 2: if floating point 3: if length 4: if angle 5: if coordinate 6: if transformation 7: if text 11: if array of integers 12: if array of floating point values 13: if array of length values 14: if array of angles 15: if array of coordinates 16: if array of transformations

A P P E N D I X

Mathematical Description

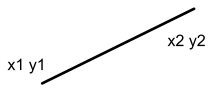


This appendix gives the exact mathematical description of the geometric objects.

Point

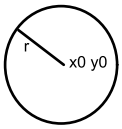
$$x(t) = x_0$$
$$y(t) = y_0$$

Line



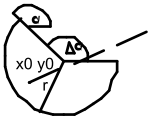
$$x(t) = t * (x_2 - x_1) + x_1$$
$$y(t) = t * (y_2 - y_1) + y_1$$

Circle



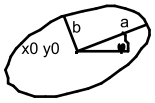
$$x(t) = x_0 + r * \cos(T)$$
$$y(t) = y_0 + r * \sin(T)$$
$$T = t * 2 * \pi$$

Arc



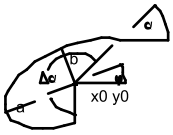
$$x(t) = x_0 + r * \cos(T)$$
$$y(t) = y_0 + r * \sin(T)$$
$$T = \alpha + t * (\alpha + \Delta\alpha)$$

Ellipse



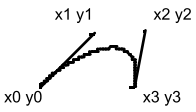
$$x(t) = x_0 + a * \cos(\varphi) * \cos(T) - b * \sin(\varphi) * \sin(T)$$
$$y(t) = y_0 + a * \sin(\varphi) * \cos(T) + b * \cos(\varphi) * \sin(T)$$
$$T = t * 2 * \pi$$

Elliptical arc



$$x(t) = x_0 + a * \cos(\varphi) * \cos(T) - b * \sin(\varphi) * \sin(T)$$
$$y(t) = y_0 + a * \sin(\varphi) * \cos(T) + b * \cos(\varphi) * \sin(T)$$
$$T = \alpha + t * (\alpha + \Delta\alpha)$$

Spline



$$x(t) = a_x * t^3 + b_x * t^2 + c_x * t + x_0$$
$$y(t) = a_y * t^3 + b_y * t^2 + c_y * t + y_0$$
$$x_1 = x_0 + c_x / 3 \qquad y_1 = y_0 + c_y / 3$$
$$x_2 = x_1 + (b_x + c_x) / 3 \qquad y_2 = y_1 + (b_y + c_y) / 3$$
$$x_3 = x_0 + a_x + b_x + c_x \qquad y_3 = y_0 + a_y + b_y + c_y$$

Where $t \in [0,1]$

APPENDIX F

Apple Events

This appendix gives the description of the available Apple Events.

topCAD supports the so called required or standard, System 7.0 friendly Apple Events and has a specific ‘topCAD friendly’ Apple Event service.

Standard Apple Events

‘odoc’ Open Documents

This Apple Event created with the standard parameters forces topCAD to open the drawings specified by the parameters. The drawing will be opened in the active foreground window.

‘pdoc’ Print Documents

This Apple Event created with the standard parameters forces topCAD to open the drawings specified by the parameters. The drawing will be opened in the foreground window. The opened document will be printed in scale 1:1 on the default printer.

‘quit’ Quit Application

This Apple Event forces topCAD to begin the standard termination process.

topCAD specific Apple Event

‘tMcr’ Execute topCAD macro

This Apple Event created with the below parameters forces topCAD to execute a topCAD macro specified by the parameters.

Event class: ‘GSTG’
Event ID: ‘tMcr’
Required Parameter:

Keyword:	keyDirectObject
Descriptor type:	typeChar
Data:	the macro instruction (e.g., /CALL “mymacro”)

Command Index

A

B

C

ℂ *(cont.)*

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

?