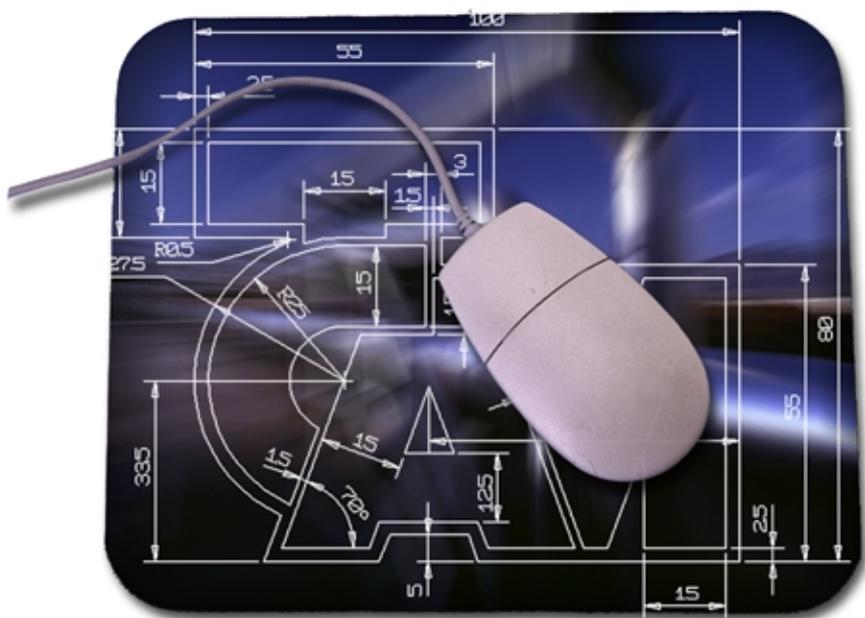


# TwinCAD<sup>tm</sup>

*Command Reference Manual V3.1*



**TCAM** 统達電腦股份有限公司  
TCAM DEVELOPMENT HOUSE CO., LTD.



# **TCAM/TwinCAD<sup>tm</sup>**

## **Command Reference Manual**

**V 3.1**

Author: Kang, Hsin Min

Copyright © 1993-1999 By TCAM Development House Co., Ltd., Taiwan, R.O.C.

TCAM is a registered trademark of TCAM Development House Co., Ltd.

TCAM/TurboCAD is a registered trademark of TCAM Development House Co., Ltd.

TCAM/TwinCAD is a trademark of TCAM Development House Co., Ltd.

TCL is a trademark of TCAM Development House Co., Ltd.

Revision Date: October, 1999

## **TCAM/TwinCAD™ Command Reference Manual V3.1**

Copyright© 1993-1999, By Kang, Hsin Min & TCAM Development House Co., Ltd.

All rights reserved. No part of this publication may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any other information storage and retrieval system, without permission in writing from TCAM Development House Co., Ltd.

### **TCAM Development House Co., Ltd.**

8F, No. 12, Lane 83, Sec.1, Guang Fuh Rd., San-Chung, Taipei Hsien, Taiwan, R.O.C.

Tel: 886-2-29990460

Fax: 886-2-29990461

Web: <http://www.tcaml.com.tw>

Email: [support@twincad.com](mailto:support@twincad.com)

**This material and the products it describes are provided in an “AS-IS” basis. TCAM Development House Co., Ltd. makes no warranty, either expressed or implied, regarding it. In no event shall TCAM Development House Co., Ltd. be liable to anyone for any special, incidental or consequential damages in connection with or arising out of the purchase or use of this material.**

TCAM Development House Co., Ltd. reserves the rights to revise and improve its products as it sees fit. This publication describes the state of this product at the time of its publication, and may not reflect the product at all times in the future.

Author: Kang, Hsin-Min

### **Trademark Acknowledgements**

TCAM is a registered trademark of TCAM Development House Co., Ltd.

TCAM/TurboCAD is a registered trademark of TCAM Development House Co., Ltd.

TCAM/TwinCAD is a trademark of TCAM Development House Co., Ltd.

TCL is a trademark of TCAM Development House Co., Ltd.

IBM is a registered trademark of International Business Machine Corp.

Windows is a registered trademark of Microsoft Corporation.

AutoCAD is a registered trademark of Autodesk, Inc.

All other brand and product names are trademarks or registered trademarks of their respective holders.

<b>Table of Contents</b>
--------------------------

3DFACE .....	MIS - 1
3DLINE .....	MIS - 2
'ABOUT .....	A - 1
ADIM .....	A - 2
ALDIM .....	A - 7
ALIGN .....	A - 14
'APERTURE .....	A - 15
ARC .....	A - 16
'AREA.....	A - 19
ARRAY.....	A - 21
ATTEXT .....	A - 24
AUDIT .....	A - 27
AUTOJOIN.....	A - 28
'AXOPLANE .....	A - 30
BASE .....	B - 1
BBREAK .....	B - 2
BISECT .....	B - 3
'BLIPMODE.....	B - 5
BLOCK.....	B - 6
BMPOUT.....	B - 10
BOX .....	B - 13
BPOLY .....	B - 15
BREAK.....	B - 18
BSPLINE.....	B - 20
BTRIM.....	B - 22
CDVIEW.....	C - 1
CENDIM.....	C - 3
'CENTROID .....	C - 4
CHAMFER .....	C - 6
CHANGE.....	C - 9
CHDIM .....	C - 12
CHINSERT.....	C - 13
CHPROP.....	C - 14
CHTEXT.....	C - 16
CIRCLE.....	C - 20
CLEANUP.....	C - 26

CLEARALL.....	C - 27
CLOAD.....	C - 28
'CMASK.....	C - 29
'CMDLIST .....	C - 30
'COLOR.....	C - 31
'CONFIG .....	C - 32
COPY.....	C - 40
COPYCLIP.....	C - 42
CSPLINE.....	C - 44
CUTCLIP.....	C - 47
DBLIST .....	D - 1
DDFRAME .....	D - 2
DDHATCH .....	D - 16
DDIM.....	D - 21
DDINSERT.....	D - 25
DDNEW .....	D - 29
DDSTYLE .....	D - 31
DDTEXT.....	D - 35
DELAY .....	D - 39
'DIMVAR .....	D - 40
DIST .....	D - 49
DIVIDE .....	D - 50
'DMODE .....	D - 53
DONUT .....	D - 55
'DOS.....	D - 56
'DRAGMODE .....	D - 58
DWGIN.....	D - 59
DWGOUT.....	D - 61
DXFIN .....	D - 66
DXFOUT .....	D - 70
'ECOLOR .....	E - 1
ELEV.....	E - 2
ELLIPSE .....	E - 3
'ELTYPE.....	E - 9
'EMODE .....	E - 10
END .....	E - 11
ERASE.....	E - 12
EXPLODE.....	E - 13
EXTEND .....	E - 15

FILES .....	F - 1
FILL .....	F - 2
FILLET .....	F - 4
FNCURVE .....	F - 7
GIFOUT .....	G - 1
'GRID .....	G - 2
HATCH .....	H - 1
HATCHEDIT .....	H - 5
HDIM .....	H - 6
'HELP .....	H - 11
'ID .....	I - 1
INSERT .....	I - 3
INVGEAR .....	I - 6
'ISOPLANE .....	I - 14
'LAYER .....	L - 1
LDIM .....	L - 3
LEADER .....	L - 4
LIMIT .....	L - 6
LINE .....	L - 7
LINETYPE .....	L - 12
LIST .....	L - 15
LOAD .....	L - 16
'LOADOPT .....	L - 18
LRDIM .....	L - 20
'LTMASK .....	L - 21
MEASURE .....	M - 1
'MEM .....	M - 3
MENU .....	M - 4
MINSERT .....	M - 5
MIRROR .....	M - 7
MOFFSET .....	M - 8
MOVE .....	M - 10
MSLIDE .....	M - 11
MULTIPLE .....	M - 12
MVCOPY .....	M - 13
NEW .....	N - 1
ODIM .....	O - 1
OFFSET .....	O - 2
OOPS .....	O - 7

'ORTHO ..... O - 8

'OSNAP ..... O - 9

'PAN ..... P - 1

PASTECLIP ..... P - 2

PASTEXT ..... P - 3

PATHCUT ..... P - 5

PCXOUT ..... P - 6

PJOIN ..... P - 7

'PLAN ..... P - 8

PLINE ..... P - 9

PLOT ..... P - 11

'PLOPT ..... P - 19

POINT ..... P - 22

POLYGON ..... P - 23

PRPLOT ..... P - 26

PTRIM ..... P - 34

'PUCS ..... P - 36

PURGE ..... P - 39

QBREAK ..... Q - 1

QCHANGE ..... Q - 2

'QTEXT ..... Q - 5

QTRIM ..... Q - 6

QUIT ..... Q - 7

RDIM ..... R - 1

'RECORD ..... R - 4

REDO ..... R - 5

'REDRAW ..... R - 6

'REGEN ..... R - 7

REGION ..... R - 8

REMOVE ..... R - 9

RENAME ..... R - 11

RESUME ..... R - 12

RHATCH ..... R - 13

RINSERT ..... R - 14

ROTATE ..... R - 16

RSCRIPT ..... R - 18

'RUN ..... R - 19

SARC ..... S - 1

SAVE ..... S - 4

'SAVEOPT .....	S - 6
SCALE .....	S - 9
'SCOLOR .....	S - 11
SCRIPT .....	S - 12
SELECT .....	S - 13
'SELMASK .....	S - 17
'SETDIM.....	S - 19
SETWIDTH .....	S - 24
'SHOWDIR .....	S - 26
SLEADER .....	S - 28
'SNAP.....	S - 29
SPLINE .....	S - 31
STRETCH.....	S - 32
STYLE.....	S - 34
'SWITCH .....	S - 37
SYMBOL .....	S - 39
SYMLIB.....	S - 43
'SYSVAR.....	S - 46
TABLET .....	T - 1
TAGEDIT .....	T - 5
TAGIN .....	T - 8
TAGOUT .....	T - 11
TAGS .....	T - 12
TAGVAR .....	T - 14
TEXT .....	T - 16
TIFFOUT .....	T - 26
'TOOLBAR .....	T - 27
TRIM .....	T - 29
'UDFVAR.....	U - 1
UNDO .....	U - 2
UNDUP .....	U - 4
UNFILL.....	U - 5
'USERVAR .....	U - 6
'VARLIST .....	V - 1
VDIM .....	V - 2
'VIEWOPT.....	V - 7
'VPOINT .....	V - 11
VSLIDE .....	V - 14
WBLOCK .....	W - 1

WMFOUT .....	W - 2
WTCAM .....	W - 3
XDIM .....	X - 1
XPDL.....	X - 4
YDIM .....	Y - 1
'ZOOM.....	Z - 1
'ZX {I,O,W,E,P,N}.....	Z - 3

# 3DFACE

## Draw 3-D Face Command

---

### Purpose:

The **3DFACE** command is used to create 3-D faces.

### Description:

The **3DFACE** command lets you create 3-D faces to model complex three-dimensional surfaces. This is an ACAD-compatible command. It is intended for applications in building 3-D objects with simple surface model.

When you enter the **3DFACE** command, **TwinCAD** will prompt in sequence:

First point:

Second point:

Third point:

Fourth point:

Third point:

A sub-command option "I" is provided for specifying that the edge starting from the next designated point will be invisible. You may continue to define a 3-D Face by specifying additional third and fourth points. The common edge carried forward from the last face will retain the same visibility attribute. You may enter space bar to the last prompt for a continuous 3-D face to exit the command.

### Procedure:

@CMD: **3DFACE** (*return*)

First point: (*pnt*)

Second point: (*pnt*)

Third point: (*pnt*)

Fourth point: (*pnt*)

Third point: (*return*)

### Example:

**3DLINE****Draw 3-D Line Command**

---

**Purpose:**

The **3DLINE** command is used to draw 3-D lines.

**Description:**

The **3DLINE** command lets you draw 3-D lines. This is an ACAD- compatible command. The **LINE** command will be called for the purpose. See also **LINE** command for details.

**Procedure:**

See **LINE** command for example procedure.

**Example:**

**'ABOUT'**

## Show Informations About TwinCAD

---

**Purpose:**

The **ABOUT** command is used to show the informations about **TwinCAD**'s version, copyright notice and other related subjects.

**Description:**

The **ABOUT** command gives you some informations about the CAD system you are using. It will pop up an information window. You may press <ESC> key or the right button of the mouse to close the information window.

**Procedure:**

@CMD: **ABOUT** (*return*)  
(*TwinCAD showing information window*)



## Angular Dimensioning Command

---

### Purpose:

The **ADIM** command is used to create an angular dimension.

### Description:

The **ADIM** lets you dimension an angle formed by selecting

- **Three points**, an angle vertex, the starting point and the ending point, forming an angle counter-clockwise. You may specify each of the start point and the end point by direct angle value.
- **Two lines**, forming an intersection angle from the first line to the second line around the intersection point, counter-clockwise. The extension lines of dimensions are also accepted for the line specifications.
- **An arc**, giving the span angle with angle vertex at center.
- **A circle**, giving the angle vertex at its center, requiring another two points on the circle to define the angle counter-clockwise.

The dimension text is generated automatically in units of degree. Dragging of the dimensioning result is provided and is started as soon as you finish the specification of the angle. The dimension lines will be inside of the angle if the dimension text is dragged inside, and will be outside, if the text is outside. The text may be generated in circular fashion with the center at the angle vertex or in horizontal direction, depending on the default option flag set by the system variable **DIMAOPT**.

During the dragging of dimensioning, you may directly enter the radius value where the dimension lines are with respect to the angle vertex, while the dragged position indicating the direction to the start point of the dimension text. Or, you may directly designate the position where the dimension text as well as dimension line should be placed. There are also several sub-command options provided here for you to modify the dimensioning result before its settlement:

- C**     **Change text**, specifying to change the dimension text manually. **TwinCAD** will prompt:

Dimension Text:

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

- D**     **Default text**, specifying to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about it.
- CO**    **Complement angle measurement**, specifying to change the current angle measurement to the one measured from the current ending angle direction to the angle direction opposite to the current starting angle direction. That is, the current

ending angle direction will become the starting angle direction, and the new ending angle direction will be taken from the opposite direction of the original starting direction. The measurement is always counter-clockwise. Note that if the dimension text has been changed manually, then the change of the measurement angle will not change the dimension text, unless you issue the Default text sub-command option.

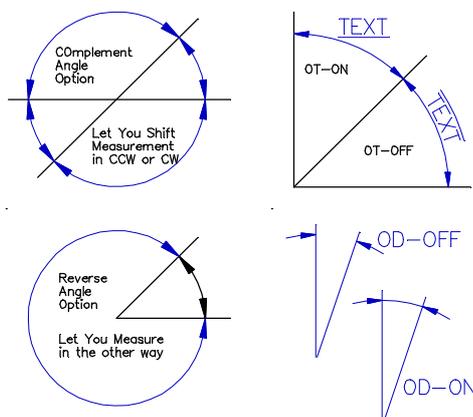
- R** **Reverse angle measurement**, specifying to change the angle measurement to the angle from the current ending angle direction to the current starting one counter-clockwise. That is, both the starting and ending angle directions are swapped, and the angle is measured again counterclockwise.
- OT** **Dimension-Text Option**, specifying to toggle the current state of dimension text option, which controls the style of text generation. If it is ON, the text will be generated in **horizontal** direction, regardless of the text position. If it is OFF, which is the default from the previous version, the text will be generated in **circular fashion** with the center at the angle vertex. This option can also be preset at the bit 1 of the dimension variable **DIMAOPT**.
- OD** **Dimension-Line Option**, specifying to toggle the current effect of dimension line option. The dimension line option is effective only when the dimension text is outside of the measured angle and so are the dimension lines generated outside of the extension lines of the angular dimension. When it is ON, an additional dimension line (actually circular arc) will be added across the measured angle while the dimension lines (with arrows) are outside of the angle. This option can be preset at the bit 0 of the dimension variable **DIMAOPT**.
- OR** **Dimension-Line Reverse Option**, specifying to toggle the current effect of dimension line reverse option, which means to reverse the dimension line with respect to the dimension text. This option can be preset at the bit 3 of the dimension variable **DIMAOPT**.
- A** **Align Option**, specifying to settle the dimension line placement, such that it can be fixed to align with a specific point or at a radius distance. **TwinCAD** will prompt:

Indicate a point to align the dimension line:

asking you to designate the alignment point. You may directly pick at an existing Angular Dimension to align with it, as **TwinCAD** will automatically issue the **ENDp** snap directive, or enter a specific radius value, to fix the dimension line position. Once the dimension line position is fixed, only the dimension text position can be dragged and changed by the cursor pointer.

You may release the fixed dimension line by entering this option again and pressing space bar to the prompt.

Effects of CO-option, R-Option, OT-option and OD-option:



Though there are dimension variables controlling the type of arrows used for the dimensioning, as well as the default suppression status for the extension lines, it is recommended to use **SETDIM** command to adjust these things, if necessary, after all the dimensioning jobs are done. Don't bother with these variables during the dimensioning activities. They are provided for other purposes, like dimensions generated by **TCL** programming. The **SETDIM** command is so designed as to make you feel free with the dimensioning activities.

However, if two lines are selected for the angular dimensioning, **TwinCAD** will automatically turn off the extension lines if the end points of the resulting dimension line (arc) should fall on the selected lines. Note also that the two lines will be extended to find the intersection point, and if they are parallel, **TwinCAD** will issue the error message as below:

No angle vertex formed.

Also, the generation of the extension lines will be optimal with respect to the two selected lines.

See also **DIMVAR** for more information about dimensioning.

### Dimension Text in Degree-Minute-Second Format

The dimension variable **DIMADPRE** controls the default text format generation for the Angular Dimensions. It specifies the number of decimal position after the decimal point to generate the text for the angular measurement value in units of degree. Additional formats for the Degree-Minute-Second system are also supported and are listed below for a quick reference:

- 10     Output degree only, as ddd~ ° ~
- 11     Output degree and minutes, as ddd~ ° ~mm'
- 12     Output degree, minutes and seconds, as ddd~ ° ~mm'ss"

Note: The character '~' (tilde) is inserted automatically to force the use of Standard Text Style, since the character code for the degree (°) symbol may conflict with the Extension Text Style.

See also **DIMVAR** for more information.

### Dimensioning on PUCS-Plane

The **ADIM** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection, and allows you to dimension an angle on the projected plane from the virtual space directly.

If the current **PUCS**-plane is active and valid for an Axonometric Projection, the angle formed by the user specification will be measured from the Projected UCS-plane, and the result of the **ADIM** command will be an Axonometric Transformation of the angular dimensioning from the Projected UCS-plane to the model space (as the same transformation done by **MVCOPY** command).

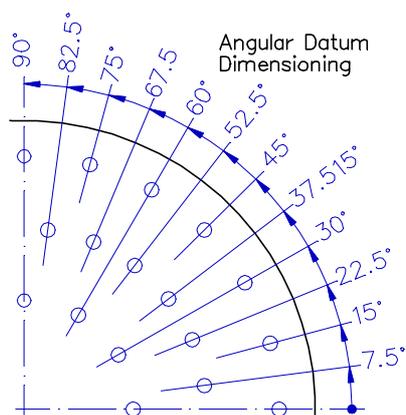
The result of such transformation will not be a single dimension entity but a dimension picture comprising of a group of elementary entities. You can not use **CHANGE** command to re-adjust such dimensioning, nor can you use **SETDIM** to modify the arrow types. So, you must set up the required dimensioning variables before doing the dimensioning under the **PUCS**-plane setup.

### Special Text Alignment Option (OA)

A special text alignment option (OA) is supported only through the dimension variable **DIMAOPT**'s bit 2, which aligns the text with the line drawn from the angle vertex to the text

position in such a manner that the Datum Dimensioning can be achieved. However, this option is not present in the command interface.

The example figure below was done by setting **DIMAOPT** to 4 (bit 2), and after dimensioning with **ADIM**, applying the **SETDIM** command to change the arrow pointers and text annotations, and suppressing the undesired extension lines. The **QCHANGE** command was also applied to trim the extension lines.



## Continuous or Base Dimensioning

If the first angle vertex point is obtained by picking at an existing Angular Dimension on one of its extension points, using **ENDp** object snap directive, **TwinCAD** will assume that you are doing the Continuous or Base Angular dimensioning, depending on which extension point is picked. You will be asked only to specify the ending angle point of the dimension, since **TwinCAD** will take the extension point you picked as the starting angle point, using the same vertex of the existing dimension.

### Procedure:

Enter the **ADIM** command to **TwinCAD** command prompt:

```
@CMD: ADIM
```

and **TwinCAD** will prompt

```
Angle vertex or <space bar>:
```

If you reply it by designating a point, you are dimensioning an angle by three points. If you reply it by pressing the space bar, **TwinCAD** will let you dimension by selecting objects. See the procedure for each case below:

- **Three points:**
  - Angle vertex or <space bar>: *(point)*
  - Starting Angle or Start Point: *(point or value)*
  - Ending Angle or End Point: *(point or value)*
  - COMplement-angle/Reverse-angle/Default-text/Change:<xxxx>/
  - OT(Text option)/OD(Dimension-line option)/<dimension position/radius>: *(Drag to position)*
- **Two lines:**
  - Angle vertex or <space bar>: *(space bar)*
  - Select arc, line or circle: *(pick up first line)*
  - Select second line: *(pick up second line)*

COmplement-angle/Reverse-angle/Default-text/Change:<xxxx>/

OT(Text option)/OD(Dimension-line option)/<dimension position/radius>: *(Drag to position)*

- **An arc:**

Angle vertex or <space bar>: *(space bar)*

Select arc, line or circle: *(pick up an arc)*

COmplement-angle/Reverse-angle/Default-text/Change:<xxxx>/

OT(Text option)/OD(Dimension-line option)/<dimension position/radius>: *(Drag to position)*

- **A circle:**

Vertex or <space bar>: *(space bar)*

Select arc, line or circle: *(pick up a circle)*

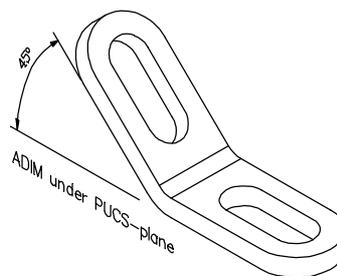
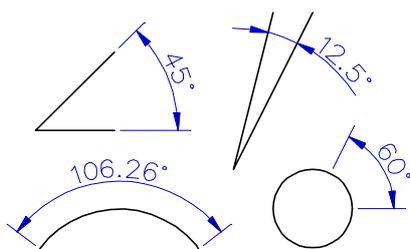
Starting Angle or Start Point: *(point or value)*

Ending Angle or End Point: *(point)*

COmplement-angle/Reverse-angle/Default-text/Change:<xxxx>/

OT(Text option)/OD(Dimension-line option)/<dimension position/radius>: *(Drag to position)*

### Example:



# ALDIM

## Aligned Linear Dimensioning Command

---

### Purpose:

The **ALDIM** command is used to create a linear dimension aligned with a given direction.

### Description:

The **ALDIM** command lets you generate a linear dimensioning which is aligned with a direction given by

- **Two points**, of which the distance is dimensioned, specifying the extension line origins.
- **A line**, of which the length is dimensioned, and of which the two end points are used as the extension line origins.
- **An arc**, of which the chord length between two end points is dimensioned.
- **A circle**, of which the diameter is dimensioned in the direction determined by the picked point. The default dimension text will be generated as that for the Diametric dimensioning.
- **An angle**, specifying the measuring direction for the above dimensioning.

You will be asked to designate the first dimension point, and then the second dimension point. If you give the first prompt a null reply, you will be asked to pick up an object (Line/Arc/Circle) for dimensioning.

The dimension text is generated automatically in the drawing unit to reflect the true length of the object. Dragging of the dimensioning result is provided and is started for you to settle its placement, as soon as you finish the basic specification of the dimensioning.

There are several ways to place the dimension text as well as the dimension lines. During the dimension dragging process, you may

- **Enter a distance value**, specifying the distance from the dimension line to the first dimension point, where the dimension text position is determined automatically by the current dragged position when the value is entered. Note that the dragged position determines only the projected position of the text to the dimension line.
- **Enter nothing but space bar**, indicating that a default distance for the dimension line should be used. The determination of a default distance depends on the mode of dimensioning. See later explanation.
- **Designate a point**, specifying directly where the dimension text should be placed. The dimension line will be generated automatically according to the setting of dimension variables (specifically, **DIMGAP**).

Before settling down the placement of the dimensioning, you may modify the dimensioning by the following sub-command options:

- D**     **Default text**, used to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information.
- C**     **Change text**, used to change the dimension text manually. **TwinCAD** will prompt:

## Dimension Text:

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

- OD** **Dimension-Line Option**, specifying to toggle the current effect of dimension line option. The dimension line option is effective only when the dimension text is dragged outside of the dimension points and so are the dimension lines generated outside of the extension lines of the linear dimensioning. When this option is ON, an additional straight line will be added inside of the extension lines, connecting the two dimension lines which fall outside of the extension lines. This option can be preset at the bit 0 of the dimension variable **DIMLOPT**.
  - OT** **Text generation option**, specifying to toggle the current effect of text generation option. It specifies whether to align the dimension text with the dimension line or not. If this option is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the same direction as the dimension line. This option can be preset at bit 1 of the dimension variable **DIMLOPT**.
  - OA** **Text alignment option**, specifying to toggle the current effect of text alignment option. It specifies whether to align the dimension text above the dimension line or not. If this option is ON, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. The dimension line will be clipped if necessary. If it is OFF, the text will be placed above the dimension line (and the dimension line will be extended to cover the span of text if necessary). This option can be preset at bit 2 of the dimension variable **DIMLOPT**.
  - OR** **Dimension-Line Reverse Option**, specifying to toggle the current effect of dimension line reverse option, which means to reverse the dimension line with respect to the dimension text. If this option is ON, the dimension line and arrows will be placed in the reverse direction. That is, if the text is inside, the arrows will be outside pointing inward. If the text is outside, the dimension line will be inside the extension lines. This option can be preset at the bit 3 of the dimension variable **DIMLOPT**.
  - O** **Oblique dimensioning**, used to generate inclined extension lines and dimension text with a specified oblique angle. The oblique angle is defined as the angle between the dimension line and a line perpendicular to the extension line. You may either use dynamic dragging function to set the dimension line or extension line direction (depending on Rotated or Non-rotated mode set by Rotated option), and the true oblique angle will be automatically calculated by **TwinCAD**; or you may directly input the oblique angle value with relative mode '@angle'.
- Note that if you enter this option right following the Rotated option, then the only parts being affected will be the dimension lines and texts; the extension lines with assigned rotated angle by previous Rotated option will not be affected. See later explanation on oblique feature and oblique angle.
- R** **Rotated dimensioning**, used to measure the object with dimension in a specific projection direction. If the measurement of the object is not to be taken from the implied direction (e.g. the length of a line from the normal direction), you may use this sub-command option to specify the projection angle explicitly. You will be asked to enter the angle value in units of degree. Relative angle value can be entered by prefixing the value with a '@' character.

To set dimensioning in Rotated mode while maintaining its original direction, enter "@0" to the angle value. To cancel Rotated dimensioning, enter nothing but null return to the angle value.

Note that entering this option will immediately cancel the Oblique dimensioning option. So, to use the Oblique feature with the Rotated dimensioning, you must enter this option first, and then oblique the dimensioning with the Oblique option.

Note also that you can not specify an effective angle direction parallel with the line passing the two dimension points for the Rotated dimensioning. This is an illegal condition, since the dimension measurement will be zero. **TwinCAD** will generate a 'beep' and cancel the Rotated dimensioning upon this situation.

- A** **Align Option**, specifying to settle the dimension line placement, such that it can be fixed to align with a specific point or at a specific dimension distance. **TwinCAD** will prompt:

Indicate a point to align the dimension line:

asking you to designate the alignment point. You may directly pick at an existing dimension to align with it, as **TwinCAD** will automatically issue the **ENDp** snap directive, or enter a specific distance value, to fix the dimension line position. Once the dimension line position is fixed, only the dimension text position can be dragged and changed by the cursor pointer.

You may release the fixed dimension line by entering this option again and pressing space bar to the prompt.

After the first dimensioning is done, **TwinCAD** will continue to prompt for the same dimensioning operation with the following added options:

- U** **Undo**, request to undo the last dimensioning.
- B** **Base**, request to generate Base Dimensioning, using the first extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimension set.

When the Base dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Base dimension mode may be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.

- C** **Continuous**, request to generate Continuous Dimensioning, using the second extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimensioning.

When the Continuous dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Continuous dimension mode may also be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.

Note that once the Continuous or Base dimension mode is entered, it will be continued automatically after each dimensioning is done. However, you may press the space bar to the prompt asking for the second dimension point to quit the mode and to go back to the first prompt where you may have other options as usual.

- H** **Horizontal**, request to switch to horizontal linear dimensioning as if you were leaving the current command and entering the **HDIM** command.
- V** **Vertical**, request to switch to vertical linear dimensioning as if you were leaving the current command and entering the **VDIM** command.
- A** **Aligned**, request to switch to aligned linear dimensioning, which is meaningless in this context as you are currently using aligned linear dimensioning.

The last three sub-command options from the above are added for easy mode switching between the Vertical, Horizontal and Aligned Linear Dimensioning. You may identify which mode you are in now by checking the parenthesized character (**H**, **V**, **A**, respectively) from the command prompt.

You have to type <Ctrl/C> to exit the command operation or give a null reply when you are asked to pick up object to put dimension.

## Default Dimension Line Distance

The determination of default distance of dimension line follows the rules below:

- If Base dimension mode is selected, the default distance will be the last dimension line distance plus or minus the default dimension line interval, depending on whether the second dimension point is outside of the extension lines of the last dimension or not.
- If Continuous dimension mode is selected, and the last dimensioning is rotated (to a given direction), the default distance will be determined such that the dimension line of the newly created one will be on the same line with that of the last one.
- If none of the above situations happen, the default distance will be the default dimension line interval specified by the dimension variable **DIMDLIR**. (see **DIMVAR** for details).

## Suppression of Extension Lines

You may use the dimension variables **DIMSE1** and **DIMSE2** to specify the default suppression status of dimension extension lines. However, it is advised not to do so, since it will slow down your productivity. It is better to use the **SETDIM** command to adjust these items after all the dimensioning jobs are done. Do not get annoyed with these variables during the dimensioning operations. They are provided for other purposes, such as dimensions generated by **TCL** programs. The **SETDIM** command is so designed as to make you feel free with the dimensioning operations.

However, for Base Dimensioning and Continuous Dimensioning, it is obvious that some extension lines would overlap as they are generated. It is therefore necessary to suppress one of them (the shorter one, of course). Sophisticated check is performed automatically to see which one should be suppressed when the dimensioning is done under these two modes. The effects of the **DIMSE1** and **DIMSE2** are ignored in these two modes.

## Virtual Segments Extension

The virtual segments generated by Dimension or Symbol entities can be accepted as valid object selections during Dimension operations. For example, you may select the extension lines and put on dimensions using the **ALDIM** command.

## Dimension Measurement Under Rotated Mode

The default dimension measurement value for a Rotated linear dimension is taken from the projected distance of the two dimension points in the Rotated direction. However, you may specify to use the true distance between the two dimension points as the dimension value in the Rotated mode, by setting the bit 7 (0x80) of the system variable **DIMLOPT** to 1.

## Oblique Feature and Oblique Angle

The Oblique feature for a linear dimension will draw its dimension line in a direction not perpendicular to its extension line. An oblique angle is thus defined as the angle between the dimension line and a line perpendicular to the extension line. This oblique feature will also

make the dimension text oblique, overriding the text oblique angle specified by the system variable **DIMOBLANG**.

The oblique angle is zero when the dimension line is drawn perpendicular to the extension line, where no oblique effect will be produced. It is positive when the dimension text should incline backward, and negative when it should incline forward. It does not matter with the actual direction of the dimension line or extension lines, since they are determined otherwise by the given dimension points or by the purpose of dimensioning.

However, to facilitate the specification of this oblique angle, the Oblique option prompts differently under the Rotated mode and Non-rotated mode.

Under Non-rotated mode, the condition is to make the extension line of the dimension be rotated to a new direction, while maintaining the dimension line at the same direction as the dimension measurement taken from the dimension points. **TwinCAD** prompts:

Oblique Angle by Extension Line Direction  $\langle nn^\circ \rangle$ :

asking you to indicate the absolute direction of the extension line. The actual oblique angle will be calculated automatically. This type of oblique dimensioning is especially useful when too many dimensions are placed in the same direction and are not easily differentiated.

Under Rotated mode, the condition is to make the dimension line of the dimension be rotated to a new direction while the direction of the extension lines are fixed by the Rotated mode. **TwinCAD** prompts:

Oblique Angle by Dimension Line Direction  $\langle nn^\circ \rangle$ :

asking you to indicate the absolute direction of the dimension line. The actual oblique angle will also be calculated automatically. This type of oblique dimensioning is used to show the effect of a parallel projection in dimensioning.

Either of the above prompts assumes a default absolute angle ( $nn^\circ$ ) that will make the effective oblique angle zero, so as to cancel the oblique feature if you reply it with a null return. You may input the angle value in relative mode ('**@angle**' format), and effectively, the relative value is the negative oblique angle.

## Oblique Arrow Pointer for Dimensions with Oblique Feature

The oblique feature of arrow pointers for the oblique dimensioning is supported and is controlled by the dimension variable **DIMOBLARW**. If this feature is turned on, the arrow pointers used for all the linear dimensioning with oblique specification will be generated with an oblique effect, suitable for presenting a 3-D projection effect.

## Dimensioning on PUCS-Plane

The **ALDIM** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection, and allows you to produce dimensions on the projected plane from the virtual space directly.

If the current **PUCS**-plane is active and valid for an Axonometric Projection, the dimension points will be measured from the Projected UCS-plane, and the **ALDIM** command will automatically calculate the required Oblique angle and set up the Rotated direction for the dimensioning as a projection result of the linear dimensioning from the Projected UCS-plane to the model space (as the same transformation done by **MVCOPY** command).

### Procedure:

Enter the **ALDIM** command to **TwinCAD** command prompt:

@CMD: **ALDIM**

and **TwinCAD** will prompt

Horizontal/Vertical/Aligned/(A)<First dimension point>:

If you reply by designating a point, you are doing dimension by two points. If you reply by pressing the space bar, TwinCAD will prompt you to select objects for doing dimensions. See the procedure for each case below:

- **Two points:**

Horizontal/Vertical/Aligned/(A)<First dimension point>: *(point)*

Second dimension point: *(point)*

ALDIM -- Default-text/Change:<xxx>/Oblique/Rotated/

OD(Dimension-line option)/<Set Dimension line position/distance>: *(Drag to position)*

- **A line, arc or circle:**

Horizontal/Vertical/Aligned/(A)<First dimension point>: *(space bar)*

Select arc, line or circle: *(pick up one)*

ALDIM -- Default-text/Change:<xxx>/Oblique/Rotated/

OD(Dimension-line option)/<Set Dimension line position/distance>: *(Drag to position)*

After the first dimensioning is done, **TwinCAD** prompts

Horizontal/Vertical/Aligned/Base/Continue/Undo/(A)<First point>:

to continue the operation.

- **With an oblique angle:**

Starting from the command prompt

ALDIM -- Default-text/Change:<xxx>/Oblique/Rotated/

OD(Dimension-line option)/<Set Dimension line position/distance>: **O** *(return)*

Oblique Angle by Extension Line Direction ( $nn^\circ$ ): *(value)*

ALDIM -- Default-text/Change:<xxx>/Oblique/Rotated/

OD(Dimension-line option)/<Set Dimension line position/distance>: *(Drag to position)*

- **With a rotated measurement direction:**

Starting from the command prompt

ALDIM -- Default-text/Change:<xxx>/Oblique/Rotated/

OD(Dimension-line option)/<Set Dimension line position/distance>: **R** *(return)*

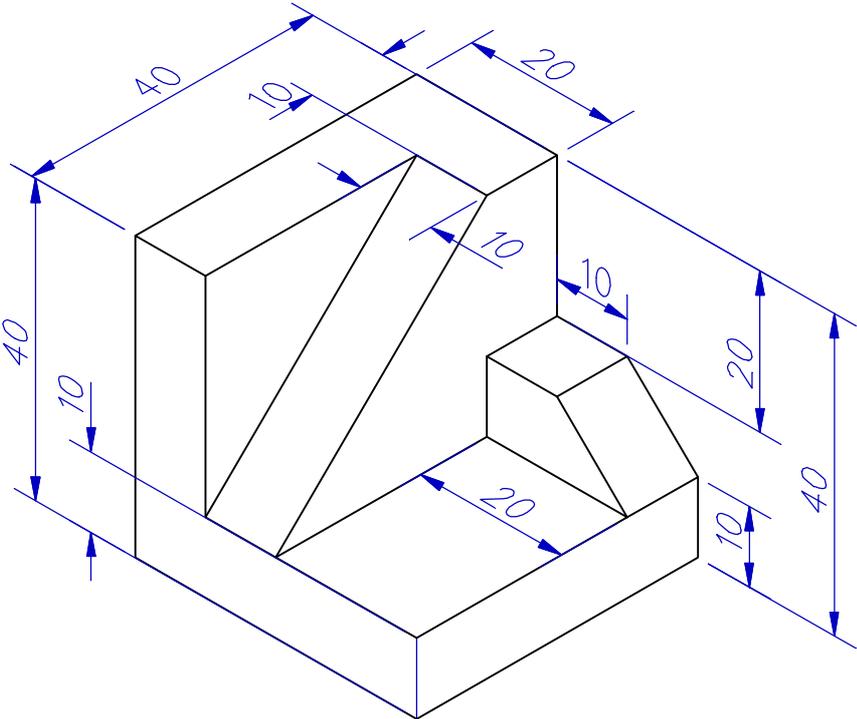
Dimension Line Direction (Rotated Angle): *(absolute direction)*

ALDIM -- Default-text/Change-text/Oblique/Rotated/OD(Dimension-line Option)

<Set dimension line position or distance>: *(Drag to position)*

### Example:

- Following are examples of dimensions generated by ALDIM.



# ALIGN

## Align Objects by Moving and Rotation Command (TCL)

### Purpose:

The **ALIGN** command is used to align selected objects to a specified position and orientation.

### Description:

The **ALIGN** command lets you move selected objects to a specified position and rotate them to align with a specified orientation simultaneously.

You will be asked first to select those objects to move and to rotate, as you enter the **ALIGN** command, by the prompt:

Select Objects (+):

After the completion of the Object Selection, you will be asked to designate points to define the base point and the destination point of the movement as well as the reference point and the target point of orientation for the rotation. As the command prompts in the following sequences:

1'st source point:

1'st destination point:

2'nd source point:

2'nd destination point:

The selected objects will be moved from the 1'st source point to the 1'st destination point and rotated about the 1'st destination point by an angle so that the orientation from the 1'st source point to the 2'nd source point will be aligned with the orientation from the 1'st destination point to the 2'nd destination point.

### Procedure:

@CMD: **ALIGN** (*return*)

Select Objects (+): (*do so*)

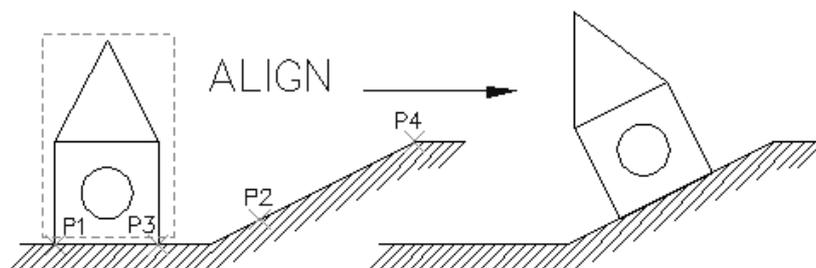
1'st source point: (*point*)

1'st destination point: (*point*)

2'nd source point: (*point*)

2'nd destination point: (*point*)

### Example:



# 'APERTURE

## Set Object Snap Box Size Command

---

### Purpose:

The **APERTURE** command is used to specify the size of the target box in object snapping.

### Description:

The object snapping box is a target box moved by cursor to pick up points from objects as you specify the object snap functions, like **INT**, **CEN**, **MID** and so on. You can specify the size of this box by this **APERTURE** command. The size of the picking box is in units of pixel (display dot unit).

You may access the **APERTURE** variable from the **SYSVAR** command to achieve the same purpose.

This is an **AutoCAD**-compatible command.

### Procedure:

Enter the **APERTURE** command to **TwinCAD** command prompt:

```
@CMD: APERTURE
```

and **TwinCAD** will prompt

```
Object snap box size in height (1-50 pixels) <7.>:
```

for you to enter the size wanted.

### Example:



## Create Arc Entity Command

---

### Purpose:

The **ARC** command is used to draw an arc segment.

### Description:

The **ARC** command is a basic drawing command, providing several methods for you to create arc segments. You may use it draw an arc segment by specifying

- **Three points**, to define an arc from the start point, through the second point to the end point.
- **A center point and two points**, to define an arc at the given center, spanning from the start point to the end point (the radius is given by the distance from the start point to center, while the end point indicates the ending angle of the arc).
- **A center point, start point and an included angle**, to define an arc at the given center, spanning from the start point with the given angle (the radius is given by the distance from the start point to center, while the sign of the angle indicates the arc direction).
- **A center point, start point and a chord length**, to define an arc at the given center, spanning from the start point to a point such that the chord length is as required (the radius is given by the distance from the start point to center, while the sign of the chord length indicates the arc direction).
- **Two points and a radius**, to define an arc starting from the first point counter-clockwise to the second point with the given radius.
- **Two points and an included angle**, to define an arc starting from the first point to the second point such that the spanning angle is as required (the sign of the angle value indicates the direction of arc).
- **Two points and a starting direction**, to define an arc starting from the first point to the second point such that the direction tangent at the first point is as required.

You may specify the start point of an arc to be the last point on the last created segment by entering the space bar to the first prompt after the **ARC** command. In such case, the tangent direction of this start point will also be determined such that the arc will tangent to the object at the start point. This tangent direction is also taken as a reference angle when you want to re-specify the starting direction by a deflection angle.

Likewise, if you specify the start point by means of the object snapping functions that return a point (e.g. **ENDp**, **MID**, **NEAr**), the tangent direction of the point on the object being snapped will also be taken as a reference angle when you want to specify the starting direction of the arc by a deflection angle. In such a case, to make the arc tangent to the object at its start point, you may specify a zero deflection angle.

All these methods are invoked by designating a point, entering a value, or typing characters for sub-command options, to the command prompt. Note that some command prompts accept both point and value, and behave differently with the type of data input.

All possible sub-command options for arc creation are listed below:

- A** - Angle, request to enter the included angle
- C** - Center, request to specify the center
- D** - Deflection, request to enter the deflection angle of starting direction
- DI** - Direction, request to enter the starting direction angle
- E** - End point, request to specify the end point
- F** - Free, request to free the specification of the starting direction
- L** - Length, request to enter the chord length
- R** - Radius, request to enter the radius

So, follow the indication of the command prompts, you may draw the arc in the method you prefer.

The **ARC** command is also invoked internally by the **PLINE** command to create an arc segment with/without a given start point and starting direction (depending whether the arc is the first segment of the polyline in creation). Additional sub-command options are added to the command prompt, and are listed below:

- S** - Spline Arc, request to draw spline arc
- L** - Line, request to draw line segment
- CL** - Close, request to close the polyline
- U** - Undo, request to undo the last segment from the polyline

See **PLINE** command for further details.

### Procedure:

Enter the **ARC** command to **TwinCAD** command prompt:

@CMD: **ARC**

and **TwinCAD** will prompt

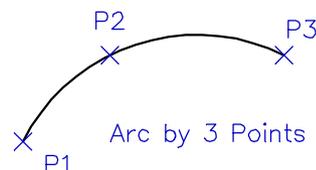
@CMD: ARC Center/<start point>:

for you to specify the start point of the arc segment or to enter the request to specify the center point of the arc first. So, different input to the command prompt leads to different case of arc creation.

See the example procedure for each case below:

- **Three points**

@CMD: ARC Center/<start point>: *(point)*  
 Center/End/Direction/Deflect/<second point>: *(point)*  
 End Point: *(point)*

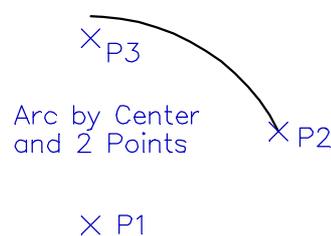


- **A center point and two points**

@CMD: ARC Center/<start point>: **C**  
 Center: *(point)*  
 Length of chord/<End point/Angle>: *(point)*

or

@CMD: ARC Center/<start point>: *(point)*



Center/End/Direction/Deflect/<second point>: **C**

Center: *(point)*

Length of chord/<End point/Angle>: *(point)*

- **A center point, start point and an included angle**

@CMD: ARC Center/<start point>: **C**

Center: *(point)*

Length of chord/<End point/Angle>: *(angle value)*

- **A center point, start point and a chord length**

@CMD: ARC Center/<start point>: **C**

Center: *(point)*

Length of chord/<End point/Angle>: **L**

Length of chord: *(value)*

- **Two points and a radius**

@CMD: ARC Center/<start point>: *(point)*

Center/End/Direction/Deflect/<second point>: **E**

End Point: *(point)*

Angle/Direction/Radius/Deflect/<Center/Radius>: *(value)*

- **Two points and an included angle**

@CMD: ARC Center/<start point>: *(point)*

Center/End/Direction/Deflect/<second point>: **E**

End Point: *(point)*

Angle/Direction/Radius/Deflect/<Center/Radius>: **A**

Included Angle: *(value)*

- **Two points and a starting direction**

@CMD: ARC Center/<start point>: *(point)*

Center/End/Direction/Deflect/<second point>: **DI**

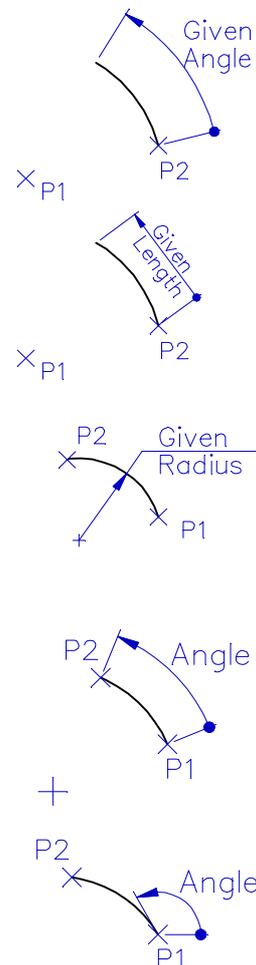
Direction angle from start point: *(value)*

or by deflection angle:

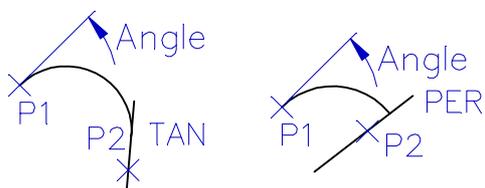
Center/End/Direction/Deflect/<second point>: **D**

Deflection Angle (+ccw,-cw)<90°>: *(value)*

End Point: *(point)*



**Example:**



**'AREA****Calculate Close Area Command (TCL)**

---

**Purpose:**

The **AREA** command is used to calculate the accumulated area of close polylines, ellipses and circles from the drawing in a simple way.

**Description:**

The **AREA** command lets you calculate the accumulated area of close polylines, ellipses and circles from the drawing in a way of successive pickups. It provides three modes of area calculation; namely, they are:

- Entity Mode** Calculate the area of single entity only. This is the initial default mode. The total area will be reset by the next single entity pickup. This mode is reset by the sub-command option "**E**".
- Add Mode** Add the area of the next single entity pickup to the total area. This mode is entered by the sub-command option "**A**".
- Subtract Mode** Subtract the area of the next single entity pickup from the total area. This mode is entered by the sub-command option "**S**".

Once the **AREA** command is entered, it will prompt

```
Area -- Add/Subtract/<Entity>:
```

asking you to pick up the entity for the area calculation. Valid entities are polylines, circles and close ellipses. If the polyline is not closed, it will be taken as a close polyline made by adding an additional line joining its start point and end point. You may enter the sub-command option "**A**" and "**S**" to switch to the Add mode and Subtract mode, respectively. **AREA** will change the prompt to

```
Area -- Entity/Subtract/<Add Mode>:
```

and

```
Area -- Entity/Add/<Subtract Mode>:
```

respectively.

Once a valid object is picked up, **AREA** will calculate the area result and report it in the command area in the form as

```
Area = xxx.xxx, Total Area = xxxx.xxx
```

and continue the prompt again, until you press space bar or <Ctrl/C> to terminate it.

**Other type of area calculation**

For complicated area formation, use **BPOLY** command to obtain the total area information.

**Special Notes:**

The **AREA** command is an external command provided by the TCL program file "AREA.TCL" or "AREA.TCA". Either file must be present in the COMMANDS sub-directory where the

**TwinCAD** resides. If you can not issue **AREA** command, you may solve the problem by copying the "AREA.TCL" to the COMMANDS sub-directory.

**Procedure:**

```
@CMD: AREA (return)
Area -- Add/Subtract/<Entity>:
...
```

**Example:**

## Copy Objects in Pattern Command

---

### Purpose:

The **ARRAY** command is used to copy selected objects in a rectangular or circular pattern.

### Description:

The **ARRAY** command lets you make multiple copies of selected objects in either a rectangular pattern or a circular pattern. Each copy of the selected objects is independent of each other, as if you make the copy by a single **COPY** command.

The object selection operation is invoked first to let you select these objects to copy. And then, you specify the type of pattern to copy by entering the sub-command options

**R** - Rectangular, request to specify a rectangular pattern

**P** - Polar, request to specify a circular pattern

to the command prompt

Rectangular/Polar array (R/P):

after the object selection.

If the rectangular option is selected, you are then required to establish the rectangular pattern by specifying the number of rows and columns, and spaces between each row and each column. **TwinCAD** will make a copy of the selected objects on each pattern point for you, assuming the originals are at the base point.

If the polar option is selected, you are then required to establish the circular pattern by specifying the rotation center and any two conditions from the three:

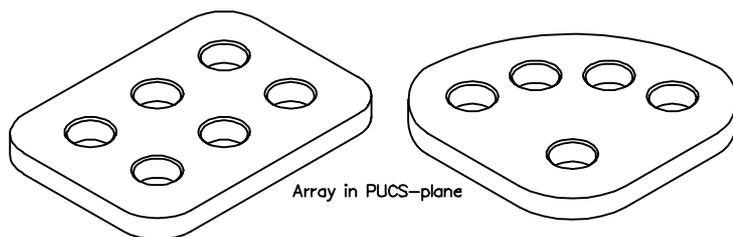
- the number of copies to make,
- the angle span to fill, and
- the angular interval between copies.



the original place first before copying. This makes the copying operation to produce insertion references to the block (block instances) instead of duplicating each of these objects, and thus saves the drawing space.

### Array Objects in PUCS-plane

The **ARRAY** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection and to create array of objects in the projected UCS-plane. However, for Polar array, the objects can not be rotated while they are copied. So, the prompt asking whether to rotate or not will be skipped over, and you will be asked to designate the reference point directly.



### Special Notes about Polar Array

In the case of polar array of objects where the angle to fill and angle between elements are given, if the angle to fill is  $360^\circ$  and is an exact integer multiple of the angle between elements, the last element at the end of the angle to fill ( $360^\circ$ ) will not be copied. This is a special case in terms of a consistent definition of the polar array copy operation, since the copying of elements to fill up a given span of angle works inclusively.

For example, if the total angle to fill is  $90^\circ$  and angle between elements is  $30^\circ$ , then there will be 4 elements copied at  $0^\circ$ ,  $30^\circ$ ,  $60^\circ$  and  $90^\circ$  respectively. In fact, the same result will be produced for a total angle value from  $90^\circ$  to  $119.999^\circ$ . So, it can be seen that the case in  $360^\circ$  is treated specially based on the assumption that the redundant one at the last copy is undesired.

Another obvious example that provokes the inconsistency issue is in the case to specify the number of element to copy in a given angle span. The same result in the above case can be obtained by specifying 4 element to copy in the span angle of  $90^\circ$ , where the angle between element is calculated as  $90^\circ/3 = 30^\circ$ . However, if the span angle is  $360^\circ$ , the angle between elements is calculated as  $360^\circ/4=90^\circ$ , so as to make the 4 copies distributing evenly over an imaginary circle, which is a result expected by the operators almost on their instinct!

So, to reason the above case for  $360^\circ$  fill and make the polar array copy operation more consistent in concept, an additional rule has been added as such:

If the last rotational copy will be redundant with the first one, a so-called Cyclic Array Copy is intended, and the copied elements will be evenly distributed around the center of rotation and satisfying the given conditions (such as number of elements or angle interval).

Note that the total span angle can be larger than  $360^\circ$ .

### Procedure:

Enter the **ARRAY** command to **TwinCAD** command prompt:

@CMD: **ARRAY**

and **TwinCAD** will prompt for you to select objects to copy first:

Select Object (+):

After the object selection, if you have selected something, **TwinCAD** will ask you to specify the array to copy, as in the following case:

- **Rectangular Array**

Rectangular/Polar array (R/P): **R**

Number of rows (---) <1>: *(Value)*

Number of columns (|||) <1>: *(Value)*

Unit cell or distance between rows (---): *(Value)*

Distance between columns : *(Value)*

Note that if either value of the row and column number is less than 1, the command quits. If equal to 1, the corresponding question for the distance in between will not be prompted.

- **Circular Array**

Rectangular/Polar array (R/P): **P**

Center point of array: *(Point)*

Number of items: *(Value or Space bar)*

Angle to fill (+=CCW, -=CW) <360°>: *(Value or Space bar)*

If both the item number and the angle to fill is zero, the command quits. If only the item number is zero (by space bar), then **TwinCAD** asks

Angle between items: *(Value)*

Or, if the angle to fill is zero, **TwinCAD** asks

Angle between items (+=CCW, -=CW): *(Value)*

After these prompts, **TwinCAD** continues to ask

Rotate objects as they are copied? <Y>: **Y or N**

And if you reply No, **TwinCAD** will continue to ask

Common reference point of objects: *(Point)*

So finishes the job.

### Example:

# ATTEXT

## Attribute Extracting Command

---

### Purpose:

This **ATTEXT** command is used to extract the attributes from the drawing entities to disk file.

### Description:

The **ATTEXT** command lets you extract the attributes tagged to entities from the drawing and write them out to a disk file in a predefined format for further processing, like BOM reporting, cost analysis, or a database inclusion transfer. This is an **AutoCAD**-compatible command.

There are three output file formats supported by this command:

- **CDF** -- Stands for Comma Delimited Format. In this format, the fields of each data record are separated by a comma (a default delimiter), and all character fields are enclosed in quotes (single quotes or double quotes). Such format is most easily processed by program written in BASIC language, and is ready to be read into database in dBASE III by the operation "APPEND FROM ... DELIMITED". Many of the database packages can read this format directly. The output file extension for this format is always ".**TXT**".
- **SDF** -- The same file format as that produced by the dBASE III operation "COPY ... SDF". The width of each field in a record is fixed. There is no need for field separators nor the string delimiters (such as quotes). It can be read into dBASE III database by the operation "APPEND FROM ... SDF". This is the default output file format. The output file extension for this format is always ".**TXT**".
- **DXF** -- The same as that supported by the AutoCAD(R) Drawing Interchange File format. However, it contains only block references and the attribute entities. The output file extension is ".**DXX**" so as to be distinct from the ordinary DXF drawing file.

Once you enter the **ATTEXT** command, **TwinCAD** will prompt

```
Tag attribute extract -- Entities/CDF/DXF/<SDF format>:
```

asking you to select the desired output file format. The default scope of the attribute extraction is from the whole drawing. However, the extraction of attributes may be selective from a part of the drawing by you entering the sub-command option "**E**" to the prompt. In such case, **TwinCAD** will allow you to select those objects of interests.

If you select the **CDF** or **SDF** format extracts, **TwinCAD** will ask for an output template file that controls the extraction of attributes and the formation of the CDF or SDF contents. This will be discussed in latter sections.

After specifying the output template file or the **DXF** format being selected, **TwinCAD** will ask you the enter the name of the extract file:

```
Extract output filename <drawing name>:
```

So the file is produced and the command exits.

### The Output Template File

An output template file contains information about what to extract from the drawing and how these extracted data are structured in the output file. In a sense, the output template file

serves as a sieve to sieve the attributes from the drawing and also as an output formatter to arrange the sieved result to the disk file in a predefined format.

An output template file is a line-oriented ASCII text file, a file that can be created and edited by using a text editor or a word processor. Each line of the file specifies one field of the extracted output record, including the name of the field, its width in character units, and the decimal position if it is a numerical field. An extracted file record is thus composed of these fields specified by these lines in their order of presence in the template file.

The format of a field specification by a text line is as below:

*Field-name* Nwwwddd

for numerical field, or

*Field-name* Cwww000

for character field, where

- **Field-name**, the name of the field, specifying the attribute name to be extracted. Only those objects tagged with the specified attributes will be extracted to the output record. The field name can be of any length, but only the first 15 characters are taken.
- **N**, Capital character, specifying numerical field intended. The content of the attribute will be evaluated as a numerical value, and output as the field format specified.
- **C**, Capital character, specifying character field intended. The content of the attribute will be copied to the field, left-adjusted, padded with spaces at the end or truncated to the field width.
- **www**, three decimal digits, specifying the field width.
- **ddd**, three decimal digits, specifying the decimal precision of the output numerical value, meaningless to character field.

The rest of the line will be ignored and can be used as the comment field.

**TwinCAD** also provides some internal attribute names that account for the extraction of the objects geometries (**AutoCAD**-compatible):

BL:LEVEL	Nwww000	(Block nesting level)
BL:NAME	Cwww000	(Block name)
BL:X	Nwwwddd	(X coordinate of insertion point)
BL:Y	Nwwwddd	(Y coordinate of insertion point)
BL:Z	Nwwwddd	(Z coordinate of insertion point)
BL:NUMBER	Nwww000	(Block counter)
BL:HANDLE	Cwww000	(Meaningless in this version)
BL:LAYER	Cwww000	(Block insertion layer name)
BL:ORIENT	Nwwwddd	(Rotation angle)
BL:XSCALE	Nwwwddd	(X scale factor of block)
BL:YSCALE	Nwwwddd	(Y scale factor of block)
BL:ZSCALE	Nwwwddd	(Meaningless in this version)
BL:XEXTRUDE	Nwwwddd	(Meaningless in this version)
BL:YEXTRUDE	Nwwwddd	(Meaningless in this version)
BL:ZEXTRUDE	Nwwwddd	(Meaningless in this version)

You may include any of them in the template file to retrieve the information you want. Note that you may replace the ':' in the **BL:xxxx** with the '\$' (dollar) or the '\_' (underscore). However, you must include at least one attribute tag field in the template file. Because, it is the attribute

tag field that determines which attribute, and hence which object is to be included in the extract file.

Once an object is determined to be included in the extract file, all the attribute fields of a record will be filled with the contents taken from the corresponding attributes tagged with the object. Should there be no corresponding one found in the object, the field will be filled with blanks.

An attribute tag field of particular name must not appear more than once in the template file. If so, only the first one is effective, and the rest will be filled with blanks.

## Special Note

**TwinCAD** allows tag attributes be appended to any kind of entities. But, to remain compatible with **AutoCAD**, this command is limited to access only the tag attributes appended to block instances.

## Procedure:

- **Extract Only Selected Entities**

@CMD: ATTEXT

Tag attribute extract -- Entities/CDF/DXF/<SDF format>: **E**

Select Object (+): *(do so)*

Tag attribute extract -- CDF/DXF/<SDF format>: *(continue)*

- **Extract in CDF/SDF format**

@CMD: ATTEXT

Tag attribute extract -- Entities/CDF/DXF/<SDF format>: **C** or **(return)**

Output template file <default>: *(file window operation)*

Extract output filename <default>: *(file window operation)*

- **Extract in DXF format**

@CMD: ATTEXT

Tag attribute extract -- Entities/CDF/DXF/<SDF format>: **D**

Extract output filename <default>: *(file window operation)*

## Example:



# AUTOJOIN

## Automatic Polyline Joining Command

---

### Purpose:

This **AUTOJOIN** command is used to create multiple polylines by joining selected segments.

### Description:

The **AUTOJOIN** command lets you make multiple polylines in one single step. All selected segments (Lines/Arcs/polylines) will be turned into polylines and joined together as possible.

You will be asked first to select all the segments to convert to polylines. Only Lines, Arcs and Polylines are valid for this operation, and **AUTOJOIN** will automatically set up the selection mask on object type for you during the selection. However, you may still need to check on the layer selection mask. See **SELECT** command for details.

After the selection, you will be asked to enter the joining epsilon value which is used to tell whether two points coincide or not. If both the delta distances (delta X and delta Y) between two points fall within this epsilon value, they meet together.

For two segments to be joined together, they must meet each other at one of their end points. Besides that, they both must be on the SAME LAYER as well as on the SAME ELEVATION.

The polylines are used to define contours, mostly. Now you may draw contours in the most convenient way, using **PTRIM** or other commands to clean it out, and then apply this command to finish the job at one step.

The **AUTOJOIN** will report the number of polylines newly produced. This number does not mean the total polyline number having been joined together, as the segments or polylines may be assimilated by another polyline during the joining operation and no new polyline will be produced for this. However, if all the selected segments are of single entities as LINE and ARC, this number should truly reflect the number of newly produced polylines.

### Special Notes

The **AUTOJOIN** produces polylines in a mass production manner. Since there may be thousands of entities involved during joining process, it is quite inefficient for **TwinCAD** to wait for the step-by-step instructions from the user. Rather, **TwinCAD** will follow the first-match rule to join all the segments in one time.

However, if in some case the user must have the control over the process details, then the **PJOIN** command may be used. The **PJOIN** command makes one polyline at a time, and prompts for user's input whenever there are branches.

Note that an open polyline that can be closed by itself will be closed after the **AUTOJOIN** operation, provided that no other segments can be jointed with it.

**Procedure:**

Enter the **AUTOJOIN** command to **TwinCAD** command prompt:

@CMD: **AUTOJOIN**

and **TwinCAD** will prompt in the following procedure:

Select Object (+): *(do so)*

Enter joining epsilon value <0.000001>: *(value or space bar)*

I am working ...

xxx new polylines produced

**'AXOPLANE'**

## Axonometric Projection Axes Setup Command

---

**Purpose:**

The **AXOPLANE** command is used to set up the projected axes directions for an Axonometric Projection.

**Description:**

The **AXOPLANE** command lets you set up the projected axes directions for a specific Axonometric Projection, which includes the Trimetric, Dimetric and Isometric projection. The axes setup for an Oblique Projection is also supported by this command as additional options.

When you enter the **AXOPLANE** command, the **PUCS** will be enabled automatically. The base coordinates of the **PUCS** in the model space will be displayed in the command prompt. You may enter point designation to change this base coordinates. **TwinCAD** will re-prompt again with the new **PUCS** base coordinates in display.

The following sub-command options are provided with this command:

- I** **Isometric**, specifying to set up the projected axes for an Isometric Projection Drawing. The **ISOPLANE** command will be called directly upon this option is entered. See also **ISOPLANE** command.
- D** **Dimetric**, specifying to set up the projected axes for a Dimetric Projection Drawing. You will be asked to specify the two axes of which the foreshortening factors are to be equal. The third axis direction will be located automatically by the definition of the Dimetric Projection. See later explanation.
- T** **Trimetric**, specifying to set up the projected axes for a Trimetric Projection Drawing. You will be asked to specify each direction of the three projected axes. See later explanation.
- O** **Oblique**, specifying to set up the projected axes for an Oblique Projection Drawing. You will be asked to specify the direction of the oblique axis. The foreshortening factor on the oblique axis will be 1.
- C** **Cabinet**, specifying to set up the projected axes for an Oblique Cabinet Projection Drawing. You will be asked to specify the direction of the oblique axis. The foreshortening factor on the oblique axis will be 0.5.
- OFF** **OFF**, clear the setup and reset **TwinCAD** to standard world coordinate system.

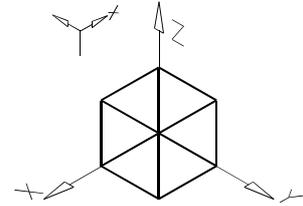
You may type <Ctrl/C> or space bar to exit the command prompt, if none of the above option is entered.

Note that Oblique Projections are not part of Axonometric Projection (they are actually different), and can not be produced by **MVCOPY** command operation; nevertheless, their setups are under the control of **AXOPLANE** command. This command is also called by the **PUCS** command.

Once the projection axes are set up, a default **PUCS**-plane using the setup will be enabled. You may use <Ctrl/I> function to toggle the **PUCS**-Plane setup.

## Isometric Projection Axes Setup

You may enter the sub-command option 'I' to set up the Axonometric Projection Axes for an Isometric drawing which has equal foreshortening factors in the three principal axes, a special case in the Dimetric Projection. The **ISOPLANE** command will be called for the axes setup. See **ISOPLANE** command for details.



If the auto-scaling function is enabled, the three foreshortening factors will be set to 1, and the value of the system variable **MVSCALE** will be set to **sqrt(1.5)** automatically. The system variable **AXOSCALE** will also be set to the same value as the **MVSCALE**.

## Dimetric Projection Axes Setup

You may enter the sub-command option 'D' to set up the Axonometric Projection Axes for a Dimetric Projection drawing which has two equal foreshortening factors in the principal axes, a special case in the Trimetric Projection.

You will be asked to specify the direction angle of the first axis with the prompt:

X-direction Angle  $\langle nn^\circ \rangle$ :

where  $nn$  is the direction angle of the X-axis of the current **PUCS**. You may indicate the direction by point designation with respect to the base coordinates of the current **PUCS**.

After that, **TwinCAD** will prompt

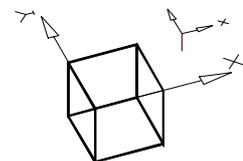
Dimetric -- Left/Top/Right/ $\langle Y$ -direction Angle ( $mm^\circ$ ) $\rangle$ :

where  $mm$  is the default direction angle of the Y-axis that has the equal foreshortening factor with the X-axis and such that the third axis will be in the vertical direction. You may enter space bar or return key to accept this default setup.

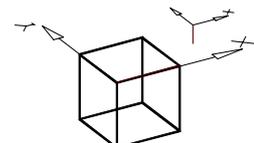
In response to the prompt, you may designate a point to indicate the second axis direction or input an angle value directly for it. The foreshortening factor for both axes will be assumed equal, and the direction and the foreshortening factor of the third axis will be determined thereby. Note that the spanning angle between the first two axes can not be zero nor  $180^\circ$ .

If the third axis is in the vertical direction, you may enter the following sub-command options to determine the second axis automatically:

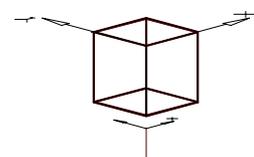
**L** **Left**, specifying that it is the second axis and the third axis that have the same foreshortening factor. The foreshortening factor of the first axis and the direction of the second axis will be calculated automatically.



**R** **Right**, specifying that it is the first axis and the third axis that have the same foreshortening factor. The foreshortening factor and direction of the second axis will be calculated automatically.



**T** **Top**, specifying that it is the first axis and the second axis that have the same foreshortening factor. The foreshortening factor of the third axis and the direction of the second axis will be calculated automatically.



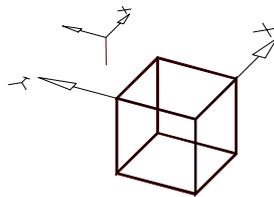
If the auto-scaling function is enabled, the foreshortening factors will be scaled up such that the two equal foreshortening factors become 1. The value of the system variable **MVSCALE** will be set to the reciprocal of the true foreshortening factor of the two axes having the same factor (since they both are set to 1). The system variable **AXOSCALE** will also be set to the same value as the **MVSCALE**.

### Trimetric Projection Axes Setup

You may enter the sub-command option '**T**' to set up the Axonometric Projection Axes for a general Trimetric Projection drawing.

You will be asked to specify the direction angle of the three projected axes, as in the following prompt sequence:

X-direction Angle <nn°>:  
Y-direction Angle <nn°>:  
Z-direction Angle <nn°>:



The exact foreshortening factors will be calculated as what they should be.

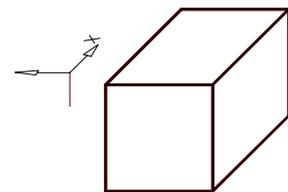
If the auto-scaling function is enabled, the foreshortening factors will be scaled up by the value stored in the system variable **AXOSCALE**, which is supposed to store the overall scaling factor for the Axonometric Projection transformation. The system variable **MVSCALE** will be reset by **AXOSCALE**.

You may use this option to set up axes for Dimetric/Isometric projection, as they are special case in Trimetric projection. However, the Auto-scaling function performs differently in determining the foreshortening factors.

### Oblique Cavalier Projection Axes Setup

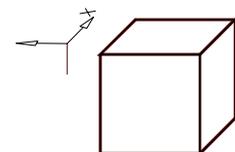
You may enter the sub-command option '**O**' to set up the Oblique Cavalier Projection Axes for Oblique Cavalier Drawing.

You will be asked to specify the direction angle of the oblique axis, which is used as the first axis. The second axis is always in the horizontal direction, and the third axis, in the vertical direction. The foreshortening factors are all set to 1. The system variable **MVSCALE** is reset to 0. This setup will not be affected by the Auto-scaling function.



### Oblique Cabinet Projection Axes Setup

You may enter the sub-command option '**C**' to set up the Oblique Cabinet Projection Axes for Oblique Cabinet Drawing. The operation is the same as the Oblique Cavalier Projection Axes setup, except that the foreshortening factor for the oblique axis is 0.5, instead of 1. This setup will not be affected by the Auto-scaling function.



You may set the required foreshortening factor in the system variable **UNITALPHA** for oblique projection drawing with other foreshortening factors.

### Auto-scaling Function in AXOPLANE Setup

The auto-scaling function in **AXOPLANE** setup will scale the length of the unit vector on the projected axis in such a way that the drawing convention is followed. Such drawing

convention was formed to make the preparing of the drawing easier to the draft-person by hand-drafting. By scaling one foreshortening factor of the projected axes to 1, and scaling the other two with the same scaling factor, the draft-person will be relieved of much of the calculation tasks in determining the measurement along the projected axes.

However, such scaling will produce side effect. The drawing prepared with such scaling is actually scaled by the same factor. It will look bigger than actually it is, when it is drawn on the same paper with its equivalent orthographic drawing.

Another side effect is that any affine transformation (such as **MVCOPY**) of an entity from the model space to the projected space must be scaled by the same factor as well; otherwise, the drawing will be out of scale. The Axonometric Projection option in the **MVCOPY** command will determine the scale factor for the affine transformation and handle this problem properly (actually, all the commands that recognize and take the advantage use of the Axonometric Projection setup should take this into consideration).

You may control the use of this Auto-scaling Function by setting the value of **AXOSCALE** system variable. To totally disable this function, set the variable to zero. Any non-zero value will enable the auto-scaling function and affect the **AXOPLANE** operations in the way as described earlier.

### Procedure:

Enter the **AXOPLANE** command to **TwinCAD** command prompt:

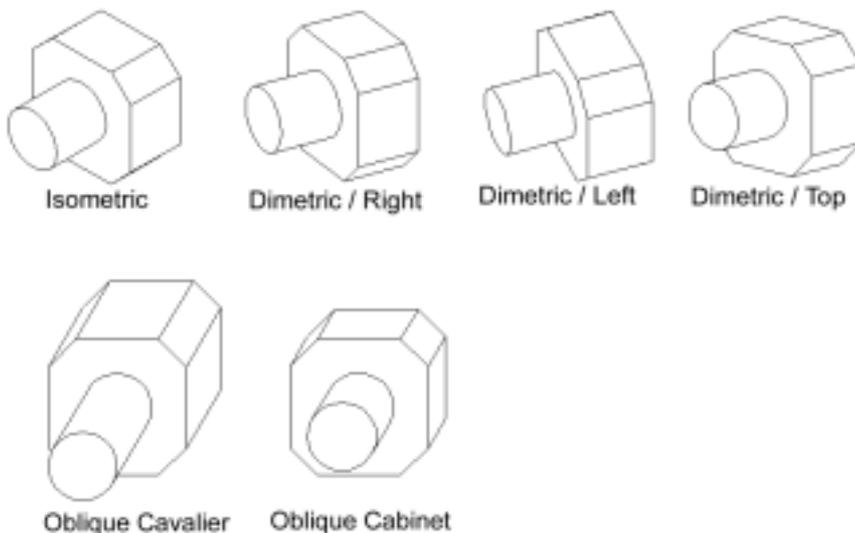
```
@CMD: AXOPLANE
```

and **TwinCAD** will prompt

```
Axoplane -- ISO/Dimetric/Trimetric/Oblique/Cabinet/OFF/<base:(0.,0.)>:
```

If you reply it by designating a point, you are changing the base coordinates of the **PUCS**. You may enter the desired sub-command option as described in the earlier section for the desired axis setup.

### Example:





**BASE**

## Set Drawing Insertion Base Command

---

**Purpose:**

The **BASE** command is used to specify the insertion origin of the drawing.

**Description:**

**TwinCAD** is capable of loading a work drawing into the current drawing in the form of block instance (a reference to block) using the **INSERT/MINSERT** command, other than simply merging it through **LOAD** command. To insert a block instance at a location in the drawing, there must be a reference point in the block of drawing to match with the location. This reference point is called the insertion base of the block. Every work drawing contains such a reference point as an insertion base, if it is to be inserted in other drawing.

The **BASE** command lets you specify the coordinates of this insertion base of the current drawing. The default coordinates of this insertion base is at (0,0), unless it is changed by the setting in the prototype drawing.

You may access the system variable **INSBASE** via the **SYSVAR** command to view or to modify this insertion base. See also **SYSVAR** command reference.

**Procedure:**

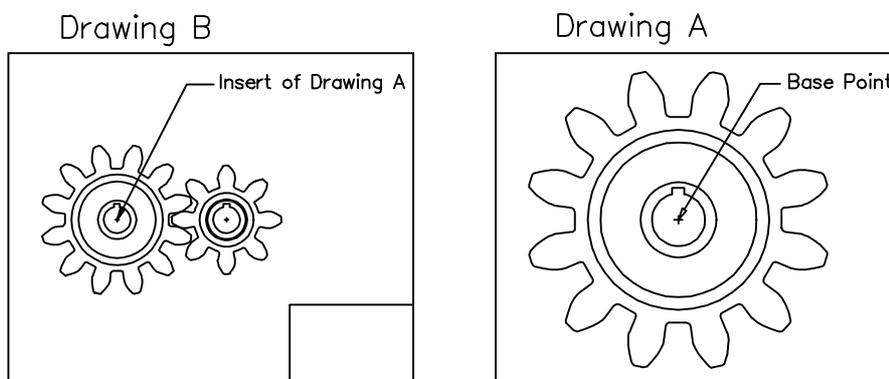
Enter the **BASE** command to **TwinCAD** command prompt:

```
@CMD: BASE
```

and **TwinCAD** will prompt

```
Drawing base point <0.,0.>:
```

for you to designate the point for insertion base.

**Example:**

# BBREAK

## Break Multiple Objects by Selected Boundaries Command

---

### Purpose:

The **BBREAK** command is used to break selected objects into parts by one or more chosen boundaries. This is an advanced drawing editing command.

### Description:

The **BBREAK** command lets you break selected objects into parts by specifying the desired boundary objects as the cutting knives. Objects that can be broken into parts are Lines, Arcs, Circles, Ellipses and Polylines.

Enter the **BBREAK** command and you will be asked to select first those objects that are to be broken into parts, as the general Object Selection Operation prompts:

Select Objects (+):

After the completion of the object selection, you are then asked to pick up the boundary objects that are to be used as the cutting knives, one by one, as the command prompts:

Un-pick/<Select Boundary Object (No.*n*)>:

where *n* is the number of boundary objects that have been picked up. Those boundary objects will be high-lighted in a different way so that you may clearly see what will be cut into parts. A selected object may also be chosen as the cutting boundary, although it may be broken into parts by other cutting boundaries.

Objects that can be chosen as the cutting boundaries are Lines, Arcs, Circles, Ellipses and Polylines. Texts can also be chosen as cutting boundaries; however, it is their text bounding boxes (with a gap distance specified by the **TXTBOXGAP** system variable) used as the cutting boundaries. At most 32 objects can be chosen in the subsequent object pickup operation. You may use the sub-command option, "**Un-pick**", to undo the last pickup of the boundary object.

To end the boundary object pickup and complete the **BBREAK** operation, enter a null reply to the prompt.

### Procedure:

@CMD: **BBREAK** (*return*)

Select Objects (+): (*do so*)

Un-pick/<Select Boundary Object (No.1)>: (*do so*)

...

### Example:

# BISECT

## Create Line/Angle Bisector Command(TCL)

---

### Purpose:

The **BISECT** command is used to create angle bisectors or perpendicular bisectors of line segments.

### Description:

The **BISECT** command lets you create an angle bisector to an angle formed by two selected line segments, or a perpendicular bisector to a single line segment.

Once **BISECT** command is executed, it will prompt in the following sequence:

Bisect -- Select line segment to bisect:

Select second line (angle bisector) or press space bar (line bisector):

Indicate location or first point of bisector line:

asking you to select one or two line segments (either of single line entity or polyline segment). If you select only one line segment (that is, you answer with a null reply to the second prompt by pressing the space bar), then you are creating a perpendicular bisector of the line segment; otherwise, **BISECT** will create the angle bisector to the angle formed by the two selected line segments. However, if the two selected line segments do not form an angle because of being parallel to each other, the bisector will be a parallel line between them at the middle.

You must designate a point to the third prompt from the above sequence, to indicate the location or the start point of the bisector. **BISECT** also uses this point to determine which bisector is meant to create from the two possible ones for the angles formed by two line segments.

After that, **BISECT** will continue to prompt the following messages:

\*\* Do not change or free the line direction constrain...

Direction/Deflect/Free/<To point/length>:

and ask you to determine the end point of the resulting bisector. You may have the following input options:

- **A value:** You may input a value to specify the length of the resulted bisector. The end point will be calculated by the length value from the start point in the direction toward the bisecting point, which is the angle vertex or the middle point of the single line segment. If the length value is negative, it will be in the opposite direction.
- **A point:** You may designate a point from which the end point will be determined by projecting it onto the bisector.
- **Space bar:** You may press space bar to accept default end point, which is the bisecting point.

During the input to the last prompt, you may drag the resulting bisector line segment as you move the mouse pointer. In fact, this is done by calling the **LINE** command for the creation of a line with a fixed start point and orientation. So, there are other effective options like "Direction", "Deflect" and "Free" appeared in the prompt. To fulfill the purpose of the **BISECT** command to create the bisector, however, you must not change the line constrain condition using these options.

Once the creation of a bisector is completed, **BISECT** will continue to cycle through the whole procedure to create the next bisector, until you press the space bar to the first prompt, or press <Ctrl/C> to terminate it.

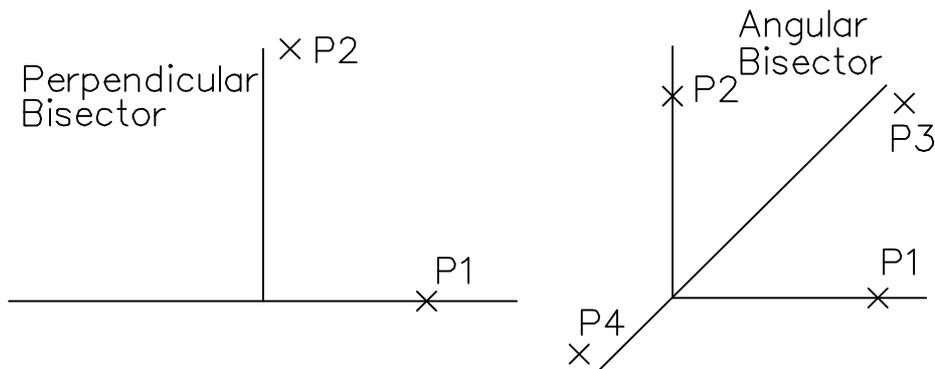
**Special Notes:**

The **BISECT** command is an external command provided by the TCL program file "BISECT.TCL" or "BISECT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **BISECT** command, you may solve the problem by copying the "BISECT.TCL" to the COMMANDS sub-directory.

**Procedure:**

- @CMD: **BISECT** (*return*)
- Bisect -- Select line segment to bisect: (*do so*)
- Select second line (angle bisector) or press space bar (line bisector): (*do so*)
- Indicate location or first point of bisector line: (*do so*)
- \*\* Do not change or free the line direction constrain...
- Direction/Deflect/Free/<To point/length>: (*do so*)
- ...

**Example:**



# 'BLIPMODE

## Turn Marker Blips ON/OFF Command

---

### Purpose:

The **BLIPMODE** command is used to turn the marker blips function ON or OFF.

### Description:

The **BLIPMODE** command lets you enable or disable the point marker blips function. This is an ACAD-compatible command.

If the marker blips function is enabled, a point marker (in small cross line) will be generated at the snapped position, whenever you pick up an object or designate a point. The marker thus generated is not part of the drawing, but helps you to identify the pickup operation. It will be removed when you redraw the screen.

You may use **DMODE** command to access the function via GUI operation. See also **DMODE** command reference.

### Procedure:

Enter the **BLIPMODE** command to **TwinCAD** command prompt:

```
@CMD: BLIPMODE
```

and **TwinCAD** will prompt

```
@CMD: BLIPMODE ON/OFF:
```

for you to enter either **ON** or **OFF** to enable or disable the function.

### Example:

# BLOCK

## Define a Named Block of Drawing Command

---

### Purpose:

The **BLOCK** command is used to define a block of drawing with a name.

### Description:

The **BLOCK** command lets you define a group of objects with a name as an entity that can be referenced and inserted at any place in the drawing. The group of objects being defined as an entity is called a block of drawing. The insertion of such entity is called a block instance (a reference to block). You may insert the block instance using **INSERT** or **MINSERT** command. (Note that the **BLOCK** is equivalent to the **SUBFIGURE** in **IGES**'s terminology.)

When you issue the **BLOCK** command, **TwinCAD** will pop up a slide window of block name selection for you to enter the name of the block you want to define. If you select an already defined block name, **TwinCAD** will assume that you are going to redefine the block. To define a new block, you have to pick up the **<NewBlock>** item from the window, and enter the name of it.

The valid name of a block must contain only the alphanumeric characters, '\$' (dollar sign), '\_' (underscore) and '-' (minus) characters. No other characters can be accepted as the block name. The letter case of the name is immaterial. Whatever you input, **TwinCAD** will convert them into upper case.

After that, **TwinCAD** will require you to designate an insertion base point of the block. The insertion base point is used as a relative origin of the objects in the block.

Then you are asked to select objects for the block definition, via the object selection operation. See also **SELECT** command.

If your drawing contains some predefined groups of attribute tags (via **TAGVAR** and **TAGS** command) and the system variable **ASKFORTAG** is not OFF, then, after the object is selected, **TwinCAD** will prompt:

Tags available. Do you want to add tags to the block ? <Y>

asking you whether to tag the block with attribute tags or not. If your answer is positive (by default), **TwinCAD** will pop up the tag group selection window from which you can choose the desired tag group to apply to the block of drawing automatically. Note that if the system variable **ASKFORTAG** is OFF, **TwinCAD** will not apply tags automatically for you using the tag group definition.

A tag group is a group of attribute tag definitions. If there is no tag group available in the drawing, then no attribute tags will be appended to the newly-defined block. You can use **TAGVAR** command to define attribute tags (including attribute name, type of tag and prompt), and then use **TAGS** command to create groups of these definitions. See **TAGS** and **TAGVAR** command reference.

Once a tag group is chosen to apply to the block of drawing, **TwinCAD** will open an attribute tag editing window for you to assign initial default values to these attribute tags. Though you may use **TAGEDIT** command to re-edit the contents of those attribute tags in later operation, it is recommended that you assign proper initial default values to them, especially those viewable tags.

For all the viewable attribute tags, **TwinCAD** will continue to prompt in sequence for each of them:

Viewable Tag: xxxxxxxxxx, Value: *(tag contents)*

Set display text -- Justify(L)/Style/<Start position>:

You may set the text justification by entering 'J', and **TwinCAD** will prompt:

Align/Fit/Even-space/Center/Middle/Right/<L>:

The default value is L(left). Note that if a tag content should be displayed with Align/Fit/Even-space justification, an initial text string must be given at the definition time (**BLOCK** command). This is because **TwinCAD** stores the alignment point implicitly using the display length of the text. Once the text length is zero, this information will be lost forever! After the justification is defined, you may change current text style by entering 'S', and continue on picking up a point as the start position to display the attribute text. The order for the viewable tags to appear is the same as that in the tag group.

If you are redefining an existing block, **TwinCAD** will check to see whether these selected objects contain references to itself. It is not allowed to define a block by referencing itself. For example, you can't define a block named TEST by an entity of **INSERT** of a block instance called TEST. Should this happen, you will be given with error messages:

Block xxxxxx already referenced in these selected objects.

Can't redefine it by these objects.

and quit the command.

When a block is redefined, the attribute tags following the block are also redefined. However, for all the existing block instances, only the constant attribute tags are redefined, while local variable attribute tags are retained.

After all these have been done, if there isn't any problem, the block is defined. Those objects you have selected will be removed logically as they have become parts of a block and, if required, attribute tags will be created to follow the block. However, **TwinCAD** will continue to ask you whether to create, automatically, a block instance (insertion reference) of the block at the original position or not. If your answer is positive, it will create such one for you so that the drawing looks the same as it was before the block definition.

**TwinCAD** will regenerate the drawing after a block is defined.

You may remove unused block definitions by **REMOVE** or **PURGE** commands.

## Commands from Script

If the command input is read from the script and the script is still active, a block name will be expected at where the dialogue window is to pop up. However, if there is no valid block name, or a backslash or a blank space is read instead, the usual block name selection window will still pop up for the user selection.

Likewise, at the Tag Group Selection operation where the user needs to select one tag group to apply to the block, **TwinCAD** will take the next word from the active script to match with existing tag groups for the selection. Only when there is no match or a backslash is read from the script for the tag group, will the usual dialogue window be pop up for the user selection. However, if a space or carriage return is read from the script instead of the name of the tag group, the selection will fail automatically and no tag group will be effective.

Note that, even though this may help the user to automatically select a tag group using the menu script, the tagging mechanism still requires the user to fill out the tags with their default values and to indicate where to put them in the drawing for those having visible attribute. It is

recommended to apply TCL function **block()** to implement automatic procedure to create specific blocks.

### Using TEXT as ATTDEF (Attribute Definition)

You may create Text entities containing special text string to define attribute tags used for Block definition. The use of such Text entities for tag attributes creation is equivalent to the ATTDEF entities in AutoCAD, which is suitable for automatic processing.

For a Text entity to be converted into equivalent tag attributes definition, it must contain the text in the format:

**&&{!#} tag-name { : initial text string }**

where

- &&** Fixed identification codes to indicate that this is a tag attribute specification rather than an ordinary TEXT entity. The text content must start with these two characters so that **TwinCAD** will check and parse its content as a tag attribute definition.
- !** Optional character to specify that the tag attribute is invisible. If this character is missing, the tag attribute will be visible by default.
- #** Optional character to specify that the tag attribute is a constant attribute. If this character is missing, the tag attribute will be variable by default. Note that it does not matter whether this character is given after the optional '!' code or before it.
- tag-name** Name of the tags. Only the first 11 characters are used.
- :** Optional character to specify that an initial default text content for that tag attribute is followed (after the ':' code).

### Procedure:

- **To define a new block**, follow the procedure
  - @CMD: **BLOCK**
  - Block name to define: *(Operate on pop-up window)*
  - TwinCAD** pops up a slide window, and you must select the **<NewBlock>** item, and enter the name of the new block.
  - Insert base point: *(point)*
  - Select Objects (+): *(Do so)*
  - Block xxxxx is defined. *(Report success)*
  - Shall an INSERT be made at the same location ? Y: *(Y or N)*
  - (Done, drawing regen)*
- **To define a new block with available tag groups**, follow the procedure
  - @CMD: **BLOCK**
  - Block name to define: *(Operate on pop-up window)*
  - TwinCAD** pops up a slide window, and you must select the **<NewBlock>** item, and enter the name of the new block.
  - Insert base point: *(point)*
  - Select Objects (+): *(Do so)*
  - Tags available. Do you want to add tags to the block ? <Y>
  - (Select a tag group from tag group selection window)*
  - (Enter initial default attribute value via window operation)*
  - (For all viewable tags...)*
  - Viewable Tag: xxxxxxxxxx, Value: *(tag contents)*

Set display text -- Justify(L)/Style/<Start position>: **J** (return)  
Align/Fit/Even-space/Center/Middle/Right/<L>: (return)  
Set display text -- Justify(L)/Style/<Start position>: (pick a point)  
Text Height <10.>: (return)  
Rotation angle or Center: (return)  
Block xxxxx is defined.(Report success)  
Shall an INSERT be made at the same location ? : (Y or N)  
(Done, drawing regen)

- **To redefine a block**, follow the procedure

@CMD: **BLOCK**

Block name to define: (Operate on pop-up window)

**TwinCAD** pops up a slide window, and you select the one to redefine from the name list.

Insert base point: (point)

Select Objects (+): (Do so)

Block xxxxx is defined.(Report success)

Shall an INSERT be made at the same location ? : (Y or N)

(Done, drawing regen)

### Example:

# BMPOUT

## Export Drawing Images in Windows BMP File Format Command

---

### Purpose:

The **BMPOUT** command is used to export the drawing image to disk file in Windows BMP file format. The image will be generated in the same way as the **PRPLOT** command prints the drawing to the printer.

### Description:

The **BMPOUT** command lets you export drawing images to disk files in the following Windows BMP file formats:

- Un-compressed monochrome image in white color background.
- Un-compressed monochrome image in black color background.
- Un-compressed 16-color image in white color background.
- RLE compressed 16-color image in white color background.
- Un-compressed 256-color image in white color background.
- RLE compressed 256-color image in white color background.

The images will be generated in the same way as you prints the drawing to the printer. However, apart from output to the printer, which has fixed resolutions and X/Y aspect ratio, printing image to bitmap file requires you to determine the output resolutions and the X/Y aspect ratio. So, the operation of the **BMPOUT** command is about the same as the **PRPLOT** command, excepts that the details of the image export configuration is different.

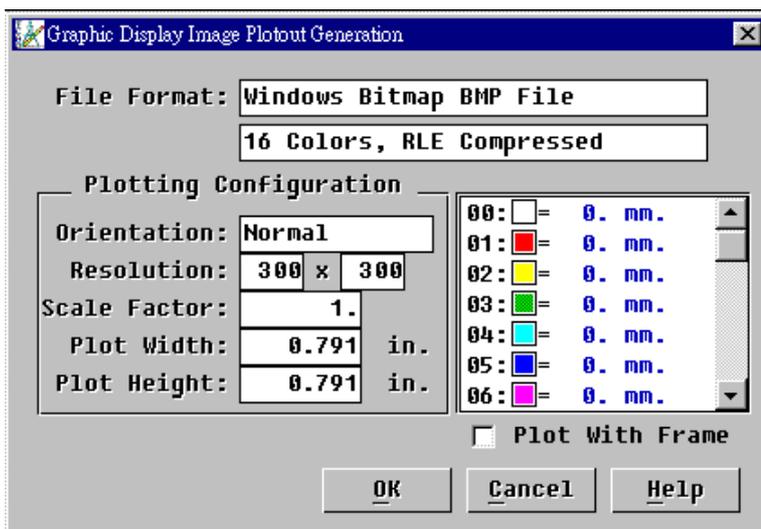
When you enter **BMPOUT** command, you will be asked first to set up a window view of the drawing to export with the following prompt:

What to plot -- Display, Extents, Limits or Window <D>:

You can enter one of the following sub-command options to specify the view window to export:

- **D - Display**, specifying the current view window to export. This is also the default if you answer it with a null return.
- **E - Extents**, specifying to use the drawing extents as the view window.
- **L - Limits**, specifying to use the drawing limits as the view window.
- **W - Window**, specifying to setup the view window by designating two points in diagonal direction. You will be asked to specify the new window as if you were issuing the **ZW** command. Only the drawing within the window defined by the two points will be scaled to export.

After you setting up the view window to export, **BMPOUT** will pop up an image export dialog window, as shown below, showing current image export configurations.



You may modify these configurations to meet your requirement and then press the [OK] button to conclude it. **BMPOUT** will pop up the file window asking you to specify the output filename and then generate the required image file if the filename is properly given.

You may quit the operation before generating the image file by pressing <ESC> key or press the [Cancel] button from the dialog window. Cancelling the file window will also quit the operation.

## Image Export Configurations

The dialog window contains the following items for you to setup before generating the image file:

- File Format** Two text fields displaying the current selected image file format name and image type. The image file format name for **BMPOUT** is always fixed as "Windows Bitmap BMP file", while you may pick at the second display field and choose to export in one of the supported image types listed before.
- Orientation** Always fixed as "Normal" in current version. If you need to rotate the image, rotate the drawing before exporting.
- Resolution** Two value entry fields containing integer values specify the image resolutions in units of DPI (dot per inch) in the X and Y direction, respectively. You must determine appropriate resolution values set to this two fields. These two values also determine the resulting image's X/Y aspect ratio. If you can not determine what resolution values are appropriate, try to set them the same as your printer's resolution.
- Scale Factor** A single value entry field specifies the scale factor of the resulting image to the original drawing, assuming the measurement unit of the original drawing is in MM. Changing this value will force the next two value entry fields, Plot Width and Plot Height, be recalculated.
- Plot Width** A single value entry field specifies the width of the resulting image in units of either MM or Inch. Changing this value will force the Plot Height and Scale Factor be recalculated. The actual width of the image in terms of pixel can be calculated by the product of this value in units of Inch and the current setting of the X resolution.

- Plot Height** A single value entry field specifies the height of the resulting image in units of either MM or Inch. Changing this value will cause the resulting image be chopped or padded with background color to the required height from the bottom.
- Image Size Unit** The image size unit can be either "mm" or "in.", after the value fields of Plot Width and Plot Height. Picking at the unit display, it will toggle from one to the other. The values in these two fields will also change as the unit changes.
- Pen Width** A scrolling sub-window containing the pen width setting used for the 16 colors. The pen width is always in units of MM. Note that the actual pen width in dot size will depend on the current setting of the image resolution. A zero pen width will cause a line to be drawn in the thinnest line width (single dot).
- Plot with Frame** A single check box to enable drawing an additional frame around the image.

### Special Notes about Image Export

The bitmap image generation depends on the capability of the Windows video driver. A device dependent bitmap (DDB) of the required size must be created first for the image generation, and then converted into device independent bitmap (DIB) format for the drawing export. It means the operation will require at least two chunk of memory blocks for the image storage.

Since the color-specification of the DDB is the same as the screen video, it means that it would take 1 bit for a pixel if it displays in monochrome, 4 bits if it displays in 16 colors, 8 bits if in 256 colors, 16 bits if in 65536 colors, and 24 bits if it displays in true colors. So the size of the memory required for the DDB creation, depends greatly on the type of the video.

The conversion from DDB to DIB is also done by the video driver through Windows API calls. The video driver also takes care of the color conversion. However, if the DDB is monochrome (i.e., the video is in monochrome), the converted color image must be bi-colors. If the DDB is in color and the required DIB is monochrome, **TwinCAD** will first convert the DDB into color DIB, and dither the DIB further into a monochrome DIB for the output using Bayer's method. This would require an additional memory chunk.

Therefore, the image export operation may fail if the image is too large and there is no sufficient memory from the system. The operation may also fail if the Windows display driver can not convert the DDB to the required DIB, especially in the RLE compression format.

Exporting image in 24-bit color format is not supported. In fact, the 16-color format is sufficient for **TwinCAD**'s drawing image export, since **TwinCAD** supports only 16 different pen colors in the drawing in current version.

Note also that although dithering technique is used to convert color image into monochrome, it is effective only to those areas of solid fill. And, certain solid fill areas like arrow heads of dimension entities are specially treated.

#### Procedure:

@CMD: **BMPOUT** (*return*)  
 What to plot -- Display, Extents, Limits or Window <D>:  
 ...[Dialog Window Operation]...

#### Example:

**BOX**

## Draw Rectangular Box Command

### Purpose:

The **BOX** command is used to draw rectangular box.

### Description:

The **BOX** command lets you create rectangular box in a much easier way.

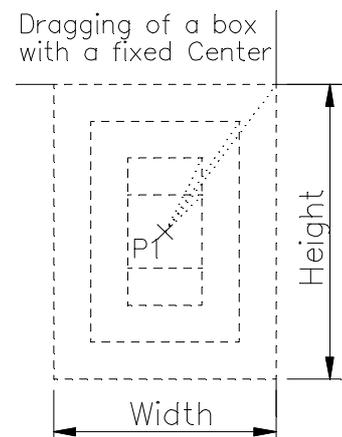
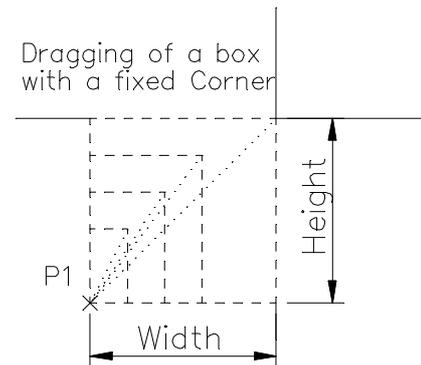
A rectangular box, consisting of 4 line segments, may be drawn by 4 successive **LINE** commands, each requiring you to enter the vertex point coordinates of the box. If the coordinates of the vertex points do not come easy (e.g. given the center position and the size of the box), you may resort to data point calculation via any possible means. This may slow down your productivity if so often boxes are to be drawn.

Depending on the data available off hand, you may draw a box by using:

- Two corner point coordinates, or
- One corner point coordinate, width and height of box, or
- One corner point coordinate, width of box, and the ratio of box height over width, or
- Center of box and one corner point coordinate, or
- Center of box, width and height of box, or
- Center of box, width of box, and the ratio of box height to the box width.

You will be asked first to designate a corner point, where a sub-command option "**C**" (Center) is provided for you to request to designate center point instead. Once the corner point or center point is given, dragging is started to create the box. Then, you will be asked to designate another corner point, or to enter the size of the box in width (recall that either data point or value can be entered to the same prompt).

If the size of box is defined by the width, you will be asked further to provide the height of the box. You may supply the height of the box in absolute value for absolute size in height, or relative value ("**@val**"), for the box's height-over-width ratio. The default height is the same as width if you reply with a null return for the height. The value entry of the width and the height of the box may be negative to indicate the negative axis direction. The resulting box will be a closed polyline.



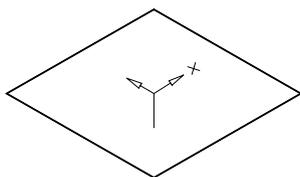
## Drawing Box on PUCS-plane

The **BOX** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to draw a box on the projected plane from the virtual space directly. If the **PUCS** mode is not active but the **ORTHO** mode is ON, the boxes drawn by the **BOX** command will still be aligned with the current **SNAPANGLE** and **SNAPBETA** setting.

### Procedure:

- **To draw a box by two corner points**  
@CMD: **BOX** Center/<Enter first point>: *(point)*  
<Another corner point/Box width>: *(point)*
- **To draw a box by one corner point, box width and height**  
@CMD: **BOX** Center/<Enter first point>: *(point)*  
<Another corner point/Box width>: *(value)*  
<Enter absolute height (*value*) or relative ratio (@hratio)>: *(value)*
- **To draw a box by a center point and ...**  
@CMD: **BOX** Center/<Enter first point>: **C**  
Box Center: *(point)*  
<Another corner point/Box width>: *(...)*

### Example:



Box Drawn Under PUCS-plane

# BPOLY

## Build Boundary Polylines Command

---

### Purpose:

The **BPOLY** command is used to create polylines of close area or a region of solid-fill area from selected boundary objects.

### Description:

The **BPOLY** command lets you create polylines of close area or a region of solid-fill area from selected boundary objects. Once it is issued, you will be asked to select those objects that may contribute to the close areas you desire. Valid objects are LINES, ARCs, CIRCLES, ELLIPSES, POLYLINES, REGIONS and non-anonymous Block Instances. See **SELECT** command for General Object Selection Operation.

It will prompt in sequence:

Select object (+):

...

Total nn single segments selected, working to define boundaries...

The message given immediately after the object selection shows the effective number of segments being taken as the boundaries to define close areas. At this moment, the command will start to build a network graph upon those effective segments. It may take a while, depends on the number of segments to scan and how fast is your machine. And thus, the number of segments reported will give you a rough idea (by experience) about how much time it will spent. Of course, if you should feel that you have selected too many objects, you may like to stop it and remove some of the objects from the selection. You may then type <Ctrl/C> to interrupt the processing and re-issue the **BPOLY** command again.

After that, the command will display the resulted network graph in the screen in blue color. You will see those close areas bounded by the network graph as loops. Each loop represents a single close area. You may **REGEN** the drawing and **ZOOM** to see the details if necessary, via the transparent commands, during the subsequent operations. Note that **TwinCAD** will draw only the network graph of connected loops at drawing regeneration during the **BPOLY** operation.

The command starts to prompt

Undo/Remove/Auto/Type/Level/<Pick a point or space-bar to end>:

asking you to designate a point to select a loop of close area. **TwinCAD** will determines the loop by the one nearest to the pick point. Once it is identified, a dash line from the pick point to the nearest segment of it will be drawn, so that you may clearly see which one you have just picked.

If you have just picked upon an already selected loop, **TwinCAD** will give you the message:

\*\* Duplicated Boundary!

and ignore the pickup. All the selected loops will be highlighted with yellow dash-line in contour or solid-fill in the loop, depending upon the current **BPOLYMODE** setting.

You may enter the sub-command option "Auto" to find the maximum area coverage from these connected loops automatically. **TwinCAD** will also identify islands and holes automatically in this Auto mode. Note that if there is no loop being selected yet before

entering this sub-command option, it will actually find out the exterior loops that cover the maximum area (which may also cover some internal connected loops) and select them accordingly. However, if there are some loops being selected, all the loops that are directly or indirectly connected with those selected loops will be bypassed during the search for a maximal coverage.

You may control the maximum level of nesting areas allowed for the loop selections by entering the sub-command option "Level". You will be asked to specify an integer value to the prompt:

Maximum Nesting Area Level <255.>:

for the current level value. See latter paragraph about the maximum nesting area level control. The setting value is stored in system variable **BPOLYLEVEL**.

If **BPOLYMODE** is ON, the selected loops will be taken as the region primitives. **TwinCAD** will use even-odd rule to determine whether a loop is an area, a hole, or an island of area, as you can see them by the solid-fill effect from the display. **TwinCAD** will calculate the total regional area and the position of the overall geometry center, and report them to the command area as:

\*\* Total Area V0=nnnn, Geometry Center P0=

If you press space-bar to end the command processing, **TwinCAD** will create a single Region entity for those selected loops of areas and give the message:

A Solid-fill Region entity is created.

However, if **BPOLYMODE** is OFF, the selected loops will be taken as independent close contours, and will be highlighted with dash-line only. No area or geometry center calculation will be done. At the end of the command, these will be converted into close polylines, and **TwinCAD** will issue:

Total nn closed polylines produced.

Before pressing the space bar to end the command processing, You may change the current **BPOLYMODE** setting by the sub-command option "Type", which allows you to choose the type of object to create with the prompt:

Type of object to create -- Region/Polyline (R):

Note that this sub-command option will not appear when the **BPOLY** operation was invoked by **HATCH**/Boundary command, since **HATCH** always assumes the target of the operation is to define a regional area. Note also that **HATCH**/Boundary will not be affected by the current setting of **BPOLYMODE** system variable, but it will definitely be affected by the setting of **BPOLYLEVEL** system variable.

During the selection of loops, you may issue the sub-command option "Undo" to un-select the last loop selection, one at a time. This is a very useful feature.

You may merge adjacent loops into a single loop by using the sub-command option "Remove", which is used to remove the segments from the network graph. **TwinCAD** prompts

Partial/Whole-seg/<Pick on boundary segment to remove (P)>:

for you to select the boundary segment to remove. There are two types of segment removing:

1. **Partial Removing**: Only the selected part of the boundary segment is removed.
2. **Whole-segment**: The whole segment (which may contribute to several loops) is removed.

Once a segment or part of a segment is removed, **TwinCAD** will trim the network graph accordingly to reflect the change. However, if a part of a boundary segment has been chosen

to define a close polyline, the whole boundary segment can not be removed. Nevertheless, in Partial Mode, if the selected part of the boundary segment is not used yet, the boundary segment will be separated into two parts and only the selected portion be removed.

The operation to remove the boundary segments works directly to modify the network graph structure internally maintained by **TwinCAD** during the **BPOLY** command processing. There is no implementation of an undo mechanism for such operation in current version and you should be careful in doing such operation.

## The Maximum Nesting Area Level Control

A single area defined by a close contour not contained by any contours that also define areas will have a nesting area level of 0. If an area is contained by another area of level 0, it will have a level of 1. Likewise, if an area is contained by other areas of which the inner most is of level N, it will have a level of N+1.

A regional area can be defined by multiple mutually non-intersectant contours. The total area is constructed by adding or subtracting each individual area defined by each contour, according to the topological relationship among them. This topological relationship is simply the nesting area level of each contour. If we said the contour starting from level 0 is the main land of the area to define, then the contours of level 1 will be lakes on the main land and the contours of level 2 will islands in these lakes. The contours of level 3 can be pools on the islands, and so on. This is how **TwinCAD** define the total area in the **BPOLY** command operation.

However, in practical application, you may need to count only the mainland area, regardless of these lakes, or the lakes are counted but not these islands, or the islands are included yet without any pools, or so and so. So as to meet these different requirements, a maximum nesting area level control is used. You can specify such a maximum nesting area level allowed for **TwinCAD** to count these areas during the **BPOLY** command operation. All the areas with a level value greater than this maximum value will be ignored. This setting value is stored in system variable **BPOLYLEVEL**.

For example, in **HATCH** command operation, if you set **BPOLYLEVEL** to 0, it will generate the equivalent effect of **AutoCAD**'s 'Ignored' hatching style. The 'Outer' hatching style can be achieved by setting **BPOLYLEVEL** to 1, and the 'Normal' hatching style, 255.

### Procedure:

```
@CMD: BPOLY(return)
Select Objects (+): (do so)
Undo/Remove/Auto/Type/Level/<Pick a point or space-bar to end>: (do so)
...
Undo/Remove/Auto/Type/Level/<Pick a point or space-bar to end>: (do so)
...
```

# BREAK

## Remove Part of An Object Command

---

### Purpose:

The **BREAK** command is used to separate an object into two parts by erasing a part of it.

### Description:

The **BREAK** command lets you erase a part of a line, arc, circle or polyline. If the erasure will split the object into two parts, it will produce two objects of the same type.

To operate this function, you must select the object to break at first. **TwinCAD** will then ask you to indicate the first point to break and then the second point. The first break point must be on the span of the object, while the second point may be outside of it, depending on your choice. Note that you can only break Lines, Arcs, Circles, Ellipses and Polylines.

If the second break point is also on the span of the object, then the portion from the first break point to the second is removed, and thus, the object is split into two parts. If the second point is outside the span of the object, the portion from the first break point to the end point nearer to the second one is removed.

If you are to split an object into two parts without erasing any part of it, reply an "@" to the second point request, or use **QBREAK** command.

### Breaking a Circle or an Ellipse

If you are breaking a circle or an ellipse, the portion from the first break point to the second point in the counter-clockwise direction is removed and the circle is changed to an arc. The break points must not be designated on the center of the circle or the arc to break, because the actual point of break is determined by the intersection of the line passing through the designated point and the center, with the circle. The same rules are also applied to breaking an ellipse.

### Breaking a POLYLINE

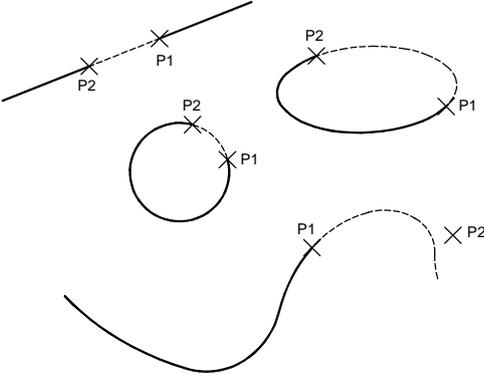
If you are breaking a polyline, you must indicate the first break point by picking on the object, using **ENDp/MID/NEAr/INT** object snap functions. The **NEAr** object snap function is assumed automatically by the request. However, the second break point is not necessary to be so indicated. If there is no object (polyline segment in this case) being picked and only a point is designated for the second break point, it is assumed that the point is outside of the span of the polyline. So, the portion that contains the polyline end point nearest to the second break point is removed. You must be aware of this rule.

### Procedure:

- **To break an object, follow the procedure**
  - @CMD: **BREAK**
  - Select object to break: *(pick one)*
  - Enter first point: *(point)*
  - Enter second point: *(point)*
- **To split an object into two parts, follow the procedure**
  - @CMD: **BREAK**

Select object to break: *(pick one)*  
Enter first point: *(point to split)*  
Enter second point: *(point)*

**Example:**



# BSPLINE

## Produce B-Spline Curve Command

### Purpose:

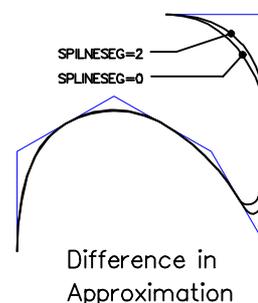
The **BSPLINE** command is used to generate a B-Spline curve over a given polyline.

### Description:

The **BSPLINE** command generates a B-Spline curve based on a given polyline. You must select a polyline that defines the desired curve.

According to the content of the system variable **SPLINETYPE**, which controls the type of B-Spline to generate, **TwinCAD** generates four types of B-Spline curves:

- 0** Continuous Quadratic Bezier Curves combined with arc segments from the original polyline. Only linear segments are converted to Quadratic Bezier curves.
- 1** Continuous Cubic/Quadratic Bezier Curve combined with arc segments from the original polyline. Only linear segments are converted to Cubic/Quadratic Bezier curves.
- 2** Non-rational Uniform Periodic B-Spline of order 3 (Quadratic). Original arc segments are not preserved.
- 3** Non-rational Uniform Periodic B-Spline of order 4 (Cubic). Original arc segments are not preserved.



The resulting curve is expressed in the form of a polyline consisting of smooth arc segments that approximate the spline, of which the fineness is controlled by the system variable **SPLINESEG** that specifies the additional number of points to divide between two given consecutive data points before the dual-arc-segment approximation is applied to each of these divisions. For example, a value of 0 in the **SPLINESEG** specifies to generate two arc segments between two given data points, a value of 1, four arc segments, a value of 2, six arc segments and so on. Should there be an inflection point in between, the number of arc segments will be doubled. Note that the inflection point, if there is any on the curve, will be located and treated as a given data point in the dual-arc-segment approximation.

Be aware that the larger value of **SPLINESEG** is, the more arc segments are produced, and then more resources are taken up to express the curve. With the dual-arc-segment approximation technique that **TwinCAD** provides, it is not necessary to have a large number of points interpolated between given data points to have a good result. Usually, a value of 1 to 2 is quite enough for most of the cases. For less important case, use the value of 0 or use the **SPLINE** command instead. See **SYSVAR** command for the method to access the system variables.

### Rules for Quadratic/Cubic B-Spline Generation

The following describes the details of the curve generation by **BSPLINE** command when **SPLINETYPE=2** and **3**:

- The polyline is used to specify the defining polygon vertices only. It does not matter with the segments (line or arc) it contains.
- If the polyline is closed, the resulting curve will be closed with the same order of continuity.

- If the number of vertex is less than 3, then no curve will be generated.
- If the number of vertex is equal to 3, then a quadratic Bezier curve will become a special case.
- If the number of vertex is larger than 3, then the uniform Periodic B-Spline curve (using uniform knot vector) will be calculated with the following boundary conditions:
  1. If the polyline is open, the starting vertex and the ending vertex will be duplicated such that the starting point and the ending point of the B-Spline curve will coincide with the two given points.
  2. If the polyline is closed, the use of vertex will be cycled around the start point (ending point) such that the resulting B-Spline curve is closed and has the same order of continuity at the starting point.

After the curve is generated, **TwinCAD** will ask you whether to erase the original polyline that defines the curve. You may choose to keep it, if you want the polyline to produce still another curve with different fineness for you.

Note that the result of the dual-arc-segment approximation will be affected by the content of the system variable **SARCTYPE**. For details, see **TCL Command Language Reference Manual** in the part about the system variable.

### Procedure:

Enter the **BSPLINE** command to **TwinCAD** command prompt:

```
@CMD: BSPLINE
```

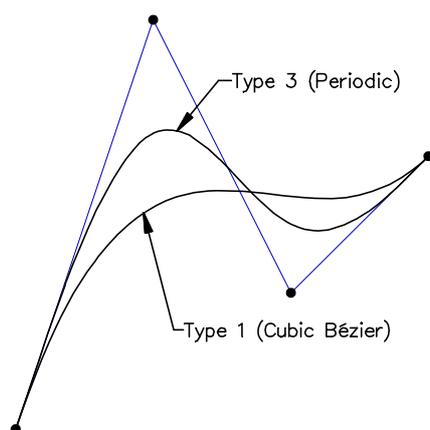
and **TwinCAD** will prompt the following

```
Select Polyline: (select one)
```

```
Done! -- New polyline with nnn entity segments created.
```

```
Delete old polyline? (Y or N)
```

### Example:



# BTRIM

## Trim/Remove Objects Based on Selected Boundaries

---

### Purpose:

The **BTRIM** command is used to trim or remove selected objects based on the drawing space partitions created by one or more chosen boundaries. This is an advanced drawing editing command.

### Description:

With some chosen boundary objects, the drawing space can be divided into several partitions. These partitions can be used to trim or to remove selected objects. So, the operation of the **BTRIM** command is to select objects, setup a drawing space partitioning, and then indicate what to trim off and what to remove, based on these partitions.

Once the **BTRIM** command is entered, you will be asked to select first those objects that are targeted to be trimmed off or to be removed in the **BTRIM** operation, as the general Object Selection Operation prompts:

Select Objects (+):

After the completion of the object selection, you are then asked to pick up the boundary objects that define the basic trimming boundaries, one by one, as the command prompts:

Un-pick/<Select Boundary Object (No.*n*)>:

where *n* is the number of boundary objects that have been picked up. These boundary objects will be high-lighted in a different way so that you may clearly distinguish them from those target objects previously selected. However, if a selected object is picked up again as a boundary object, it will be de-selected from the trimming targets, since no boundary objects can be trimmed off or removed during the BTRIM operation.

A boundary object can be a single line, arc, circle, ellipse or a complicated polyline. The text entity can also be chosen as a boundary object; however, it is the text bounding box (with a gap distance specified by the **TXTBOXGAP** system variable) to be used as the boundary. The **BTRIM** will accept at most 32 boundary objects in the subsequent pickup operation. A sub-command option, "**Un-pick**", is provided for you to un-pick the last pickup of the boundary object.

To end the boundary object pickup, enter a null reply to the prompt.

Those boundaries objects shall divide the whole drawing space (2-D) into several partitions. You are asked to indicate which particular partition is of interest to perform the trimming operation, as the command continues to prompt:

Trim-only/Remove-all/<Indicate side to trim off or to remove objects>(R):

Once a particular partition is specified, those target objects that span across it will be trimmed off by their parts that fall within it. If the [**Remove-all**] option is enabled, those that fall completely within this partition will also be removed.

If there is only one single object picked up as the boundary, the BTRIM will terminate automatically once a partition is indicated and the trimming operation is completed. This is because a single object always divides the drawing space into two partitions, and the purpose of the operation is to trim off the objects by either one of them.

However, if multiple objects are selected as the boundaries, then there will be more than two partitions. The **BTRIM** command will thus continue the same prompt for the next partition indication. At this time, a sub-command option, "**Undo**", is provided without an explicit option prompt, for you to undo the last trimming operation.

To end the trimming operation, enter a null reply to the prompt.

## Rules and Definitions:

### Space Partitioning -- Explicit Partitions vs. Implicit Partitions

An explicit partition is a partition that has an explicit area bounded by those boundary object's segments, while a partition that exhibits no explicit area bound is an implicit partition. An explicit partition is also called a close partition; however, an implicit partition is not necessary an open partition. In fact, an implicit partition is defined as a partition having invisible boundaries, either partially or totally. It is therefore possible to have a close but implicit partition.

For example, a single circle divide the space into two partitions, one is inside of the circle which is bounded by the circle, and the other is outside of the circle without bound. The one inside the circle is an explicit partition, and the one outside, an implicit partition.

Another example is a single line that divides the drawing space into two implicit partitions. There are invisible boundaries obtained by extending the line on its both sides to the infinity.

Multiple boundary objects may form complicated partitioning of the drawing space. Some are explicit partitions and some are not. As the drawing space is an unlimited space to infinity, there are always implicit partitions formed by these boundary objects. To simplify the partitioning result and to make it more visualizable to the operator, additional rules about the partitioning are implemented.

### Rules of Partitioning

Those close areas bounded explicitly by the boundary objects are located first for the Explicit Partitions. This should be visible and obvious to the operator.

Once these close and explicit partitions are found, the rest of the drawing space is then further partitioned by those boundary objects with extended geometries which contribute to invisible boundaries.

### Objects with extended geometry

Lines, arcs, elliptic arcs and open polylines are objects with extended geometry. The extended geometry of these objects provide invisible boundaries for the drawing space partitioning. A line is extended on it both sides to the infinity and an arc or elliptic arc is extended to make it a whole circle or ellipse. As for an open polyline, it depends on the type of segments on its both ends and will be very complicated.

Circle, ellipse and close polyline do not have extended geometry.

### Trimming and Removal of Objects

A targeted object will be trimmed off by the part in the specified partition if and only if it spans across that partition's boundary. This shall exclude the condition that it spans just right at the partition's boundary, since the trimming will result in either a zero length object or nothing being changed. So, if an object is completely inside a specified partition but tangent to or

touching at that partition's boundary, it will not be trimmed off anyway. It can only be removed under the [Remove-all] option.

The property of the resulting objects after trimming will remain the same as their original objects. However, a circle will be trimmed into piece of arcs.

Only Lines, Arcs, Ellipses, Circles, and Polylines can be trimmed. All other types of object will be retained or removed, depending on whether one is totally within the specified partition under the [Remove-all] option.

### Procedure:

@CMD: **BTRIM**(return)

Select Objects (+): *(do so)*

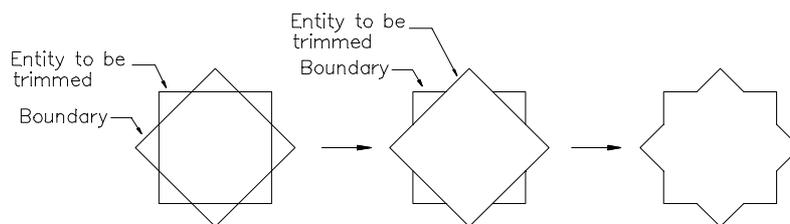
Un-pick/<Select Boundary Object (No.1)>: *(do so)*

...

Trim-only/Remove-all/<Indicate side to trim off or to remove objects>(R): *(do so)*

...

### Example:



# CDVIEW

## Create Clipped Details View Command (TCL)

---

### Purpose:

The **CDVIEW** command is used to create a clipped and magnified details of a particular view of the drawing automatically.

### Description:

The **CDVIEW** command lets you create a magnified details of a particular view of the drawing, clipped to a specified rectangle or circle. Such a magnified drawing view is useful in providing dimension details.

When you enter the **CDVIEW** command, it will prompt

Clipped Details View -- Circle-type/<Box-type>:

asking you to determine the shape of the clipping frame for the drawing details first. You may enter the sub-command option "C" to choose a circle frame or pressing space bar for a default rectangle frame. If you designate a point to this prompt, you choose the rectangle frame. The point will be taken as a corner point of the rectangle frame.

Depending on the type of the clipping frame you have chosen, **CDVIEW** will issue either one of the following messages:

\*\* Please box the part of drawing for details view.

or

\*\* Please circle the part of drawing for details view.

and then call either the **BOX** or **CIRCLE** command to create the required clipping frame.

After the creation of clipping frame, **CDVIEW** will continue to prompt

Details View -- <Scale(1.000)>/<View Center Location>:

asking you to specify the required magnification factor of the drawing and the location where the magnified result will be placed. **CDVIEW** will continue the prompt until you press the space bar to consent the current setup of the resulting frame. In the mean time, a dragging frame is provided and the current placement of the frame will also be displayed in the drawing. Both of them will be changed as the current magnification factor is changed.

When you finally press the space bar, **CDVIEW** will start creating the magnified drawing view in the designated place and clip it against the resulting frame. When it finishes, the command ends.

### Limitations

**CDVIEW** will duplicate and scale the drawing entities that fall within or crossed by the clipping frame to the new location, and then make a boundary trimming of these objects against the clipping frame. However, since there are entities not trimmable, such as Texts, Dimensions, Inserts and Symbols, you may have to modify the resulting view by yourself if necessary.

### Special Notes:



# CENDIM

## Create Center Line/Mark Command

---

### Purpose:

The **CENDIM** command is used to create a center line/mark on a circle or arc.

### Description:

The **CENDIM** command lets you create a dimension entity called Center Line/Mark, which is used to mark the center of a circle or an arc.

A Center Mark is a single cross mark of a specific size located at the center of a circle (though the circle may not exist), to indicate the position of a center. A Center Line is an extended Center Mark which has four extension arms (from the single cross) added to intersect with the circle (see the figure in Example).

You will be asked first to pick up the circle or arc for which the center mark is to be created. If you want to create the center mark on a non-existing circle, you may enter the sub-command option "**C**" to signify that request and then you will be asked to specify the center position as well as the size of the non-existing circle.

After the circle has been specified, you will be asked to enter the size of the Center Mark. If the size of center mark is negative, then you are specifying a Center Line, and you will be asked further for the rotation angle of the Center Line. The default angle is zero.

The size of the center mark, once generated, can be modified later on using the **QCHANGE** command.

### Dimensioning on PUCS-Plane

The **CENDIM** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to create a center mark on the projected plane from the virtual space directly. However, to create such a center mark, you must use the sub-command option '**C**' or pick up an ellipse that is a projected result of a circle from the virtual space under current PUCS-plane setup. The result of such center mark is still a single Dimension Entity.

### Procedure:

- **To create center mark or center line on a circle**
  - @CMD: **CENDIM** -- Select arc/circle: *(pick one)*
  - Enter center dash length (+:mark,-:line/mark)(*val*): *(value)*
  - If the value entered is negative, then the prompt continues:
  - Rotation angle <0°>: *(point or value)*
- **To create center mark or center line without a circle**
  - @CMD: **CENDIM** -- Select arc/circle: **C**
  - Center: *(point)*
  - Diameter/<Radius:(0.)>: *(point or value)*
  - And the rest are the same ...

### Example:

**'CENTROID****Calculate Centroid of Close Areas Command (TCL)**

---

**Purpose:**

The **CENTROID** command is used to calculate the accumulated area and overall centroid coordinates of close polylines, ellipses and circles from the drawing in a simple way.

**Description:**

The **CENTROID** command lets you calculate the accumulated area and overall centroid coordinates of close polylines, ellipses and circles from the drawing in a way of successive pickups. It provides three modes of centroid/area calculation; namely, they are:

- Entity Mode** Calculate the centroid and area of a single entity only. This is the initial default mode. The total centroid and area will be reset by the next single entity pickup. This mode is reset by the sub-command option "**E**".
- Add Mode** Add the area of the next single entity pickup to the total area. The overall centroid is also calculated. This mode is entered by the sub-command option "**A**".
- Subtract Mode** Subtract the area of the next single entity pickup from the total area. The overall centroid is also calculated. This mode is entered by the sub-command option "**S**".

Once the **CENTROID** command is entered, it will prompt

Centroid -- Add/Subtract/<Entity>:

asking you to pick up the entity for the area and centroid calculation. Valid entities are polylines, circles and close ellipses. If the polyline is not closed, it will be taken as a close polyline made by adding an additional line joining its start point and end point. You may enter the sub-command option "**A**" and "**S**" to switch to the Add mode and Subtract mode, respectively. **CENTROID** will change the prompt to

Centroid -- Entity/Subtract/<Add Mode>:

and

Centroid -- Entity/Add/<Subtract Mode>:

respectively.

Once a valid object is picked up, **CENTROID** will calculate the total area and overall centroid result and report them in the command area in the form as

Area V0= xxx.xxx, Centroid P0= xxxx.xxx

and continue the prompt again, until you press space bar or <Ctrl/C> to terminate it.

As the area and centroid report message implies, the resulted area and centroid are stored in system register V0 and P0, respectively. This would help you in successive operations that may require these two values.

**Other type of area/centroid calculation**

For complicated area formation, use **BPOLY** command to obtain the total area and centroid information.

### Special Notes:

The **CENTROID** command is an external command provided by the TCL program file "CENTROID.TCL" or "CENTROID.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **CENTROID** command, you may solve the problem by copying the "CENTROID.TCL" to the COMMANDS sub-directory.

### Procedure:

```
@CMD: CENTROID (return)  
Centroid -- Add/Subtract/<Entity>:  
...
```

### Example:

# CHAMFER

## Make Chamfer Between Objects Command

---

### Purpose:

The **CHAMFER** command is used to create a chamfer between two selected objects.

### Description:

The **CHAMFER** command lets you trim two intersecting objects at a given distance from their point of intersection and create a new line entity that connects the two trimmed ends, and thus make the chamfer. Different chamfer distance for each object is possible.

Objects that can be selected to make chamfers are Lines, Arcs, Polylines and Circles. However, circles will not be trimmed from the chamfering operation. The **CHAMFER** command does not require that the selected objects physically in contact, as long as the extended portions of them will intersect.

You can make chamfers over a polyline by selecting the sub-command option "**P**". The polyline selected will make chamfers for each corners automatically. You may change the chamfers of a polyline by giving new chamfer distances, and then select it again. Or, you may remove the chamfers by supplying a zero chamfer distance. The chamfer line in a polyline assumes a special attribute, so that it can be recognized and modified by the subsequent **CHAMFER** or **FILLET** commands. Note that the chamfering sequence over a polyline is always the same as that of the polyline, though the chamfer distances may not be the same.

You may change the chamfer distance by directly entering the value to the first command prompt. The value will become the chamfer distance for both objects. If you want to have different chamfer distance for each object, enter the sub-command option "**D**". **TwinCAD** will let you enter different chamfer distance for each object.

The following lists the valid sub-command options:

- (*value*) - Chamfer distance, specifying both chamfer distances
- D** - Distance, request to enter both chamfer distances
- P** - Polyline, request to chamfer over a polyline
- U** - Undo, request to undo last chamfer operation

To exit the command, type <Ctrl/C> or press the space bar.

The system variables **CHAMDIST1** and **CHAMDIST2** store the two current chamfer distances respectively. You may access these values by the **SYSVAR** command.

### Special Notes

Note that the newly created line segment of chamfer will assume the default properties from the current layer, linetype and color, provided that the chamfer is made on two separate entities. It will assume the same properties from the entities that produce the chamfer if and only if the entities are two consecutive segments from the same polyline, since the line segment of chamfer will also be included in the polyline.

Also note that a chamfer can be produced if and only if the two selected entities intersect, since the chamfer distance is calculated from the point of intersection. So, you can make

chamfer over two 3D-lines, if you are sure that they do intersect in the 3D-space somewhere. But, if the two selected entities do not intersect, you definitely will receive the error message:

Can't have such chamfer.

Common mistakes occur when you try to chamfer two entities of different elevations.

Another case for you to receive the above error message is when the chamfer will make either one of the selected entities to disappear. This does not include the case that just makes the entity become zero length (the chamfer will be accepted but the zero length entity will be removed). Also note that the Ellipse entity can not be chamfered, unless the chamfering distance is zero.

If the option "P" is used to make chamfers over a polyline, the new created chamfer line segments will internally assume a special flag, which is valid only for the current editing session. Both the **CHAMFER** and **FILLET** commands recognize this special flag and ignore the segments when the operator issues the command to make chamfers or fillets over the same polyline with the "P" option again. This allows the operator to change the chamfer distance of all chamfers or fillet radius of all fillet segments, or to replace all existing chamfers with fillets and vice versa. However, once the drawing is saved and loaded later again for the next editing session, the segments created by **CHAMFER** or **FILLET** commands will be no different from other ordinary line/arc segments.

## Procedure:

Enter the **CHAMFER** command to **TwinCAD** command prompt:

@CMD: **CHAMFER**

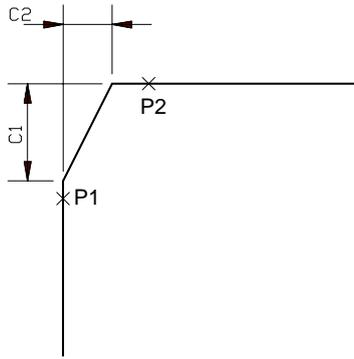
and the system will prompt first

Undo/Polyline/Distance:<d1,d2>/<Select first object>:

for you to make chamfer on selected objects. Follow the procedures below:

- **To specify both chamfer distances**  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>: *(value)*  
 Undo/Polyline/Distance:<value,value>/<Select first object>:
- **To specify each chamfer distance**  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>: **D**  
 Enter first chamfer distance<d1>: *(value1)*  
 Enter second chamfer distance<d2>: *(value2)*  
 Undo/Polyline/Distance:<value1,value2>/<Select first object>:
- **To make chamfer on two objects**  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>: *(pick object)*  
 Select second object: *(pick object)*  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>:
- **To make chamfer over a polyline**  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>: **P**  
 Select polyline: *(pick a polyline)*  
 Undo/Polyline/Distance:<d1,d2>/<Select first object>:

**Example:**



The Chamfer Distances

# CHANGE

## Modify Objects Command

---

### Purpose:

The **CHANGE** command is used to modify the properties or geometries of selected objects in the drawing.

### Description:

The **CHANGE** command lets you modify the properties (layer, color, etc.) or the geometries of the selected objects in the drawing, globally or individually.

Firstly, you are asked to select the objects to modify via the object selection operation (see **SELECT** command), and after the selection, **TwinCAD** will prompt

LType/Color/Layer/Elevation/Thickness/ALL-same/<Change Point>:

You may specify the subject to change by entering

- a point designating the change of geometry to the objects,
- a null return to change the objects by default, or
- an option character to change specific property of objects.

These will be discussed in the following paragraphs.

### Change Geometry to New Point

If you reply to the prompt by designating a point, called the change point or CP for convenience, the geometries of selected objects will be changed as described below:

**Line** One of the end points closest to the CP will be changed to CP.

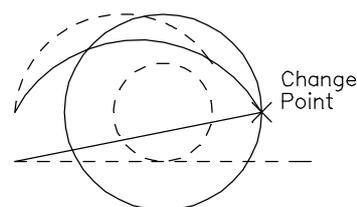
**Circle** Radius of the circle will be changed so that the CP lies on the circumference of the circle.

**ARC** One of the end points closest to the CP will be changed to CP. The arc segment is stretched so that the distance from the midpoint of the chord to the arc is held constant. The center point and the radius are adjusted as required.

**TEXT** Text insertion point is changed to CP. **TwinCAD** will continue asking you to modify the text style, text height, writing angle and text string further.

**SYMBOL** Replacement of different symbol is possible. The CP is meaningless here.

**DIMENSION** Placement of dimension will be dragged again to a new location. The CP is meaningless here, as the dragging of the dimension is restarted. The dimensioning parameters (such as options) of the dimension will be modified by the current dimensioning setup, except the dimension text,



scale and the arrow types. These will be preserved. You may change them individually by **SETDIM** command.

**REGION** The **RHATCH** command is called internally to modify the hatching state of the region. The CP is meaningless here. Although, this may be a redundant command support, but the use of **CHANGE** command to modify existing property of objects is more conceptually acceptable. Also, the use of **CHANGE/ALL**-same option is very useful in unifying the hatch patterns in those selected regions.

**Else** Nothing happens.

If different type of objects are selected, they will be changed one by one with the same change point.

Note that the number of entities being selected to change with a change point will be compared to the content of the system variable **MAXCHANGE**. If it should exceed the current setting value of **MAXCHANGE**, **TwinCAD** will prompt

\*\* More than n (m) entities are subjected to change. Are you sure? <N>:

asking you to confirm the operation, where *n* is the current setting value of **MAXCHANGE**, and *m* is the number of selected entities.

## Change Geometry without Change Point

If you reply to the prompt by a null return, **TwinCAD** assumes that you want to change the geometries of the selected objects without a point of change. So, it affects only the **TEXT**, **DIMENSION**, **SYMBOL** and **REGION** entities. See the paragraphs above.

## Change Properties by Entering Option

If you reply to the prompt by typing an option character from the following, you may change the properties of the selected objects:

- LT** Linetype, specifying to change the linetype of the selected objects. **TwinCAD** will pop up a slide window of linetype selection for you to pick up the linetype you want to change them to. If the menu script is active, a linetype name may be read directly from the script.
- C** Color, specifying to change the color of the selected objects. **TwinCAD** will pop up a slide window of color selection for you to pick up the color you want to change to. If the menu script is active, a color number may be read directly from the script.
- L** Layer, specifying to change the layer of the selected objects. **TwinCAD** will pop up a slide window of layer selection for you to pick up the layer you want to change them to. If the menu script is active, a layer name may be read directly from the script.
- E** Elevation, specifying to change the elevation of the selected objects. You will be asked to enter the new elevation value from the keyboard.
- T** Thickness, specifying to change the thickness of the selected objects. You will be asked to enter the new thickness value from the keyboard.
- ALL** Effective only to **CIRCLE**, **DIMENSION**, **TEXT**, **REGION** and **SYMBOL** entities, specifying to change once for all of the same kind. For example, when a circle is changed to a new size, then all the rest of the selected circles will assume the same new size. Another example is when a **TEXT**'s style is changed to a new one, then all the rest of the selected **TEXT**s will assume the same new style automatically. This option is quite helpful for global changing the text styles, symbols and hole diameters.

## Apply CHANGE/ALL to Dimension Entities

You may 'stretch' a group of linear dimension entities to new positions to make room for new dimensionings, using the **CHANGE** command with the **ALL** sub-command option. The rules about this operation are described below:

- For each kind of the Linear Dimension (Horizontal/Vertical/Aligned), the command will allow the user interactively change the first dimension entity encountered from the selection list individually as usual. The command will then remember an offset distance that the change has been made on its first dimension extension point.
- For the same kind of dimension entities subsequently encountered from the selection list, the command will directly add this offset value to both dimension extension points without asking for the user interaction. The dimension text position is also changed by the same amount of offset distance.
- For Aligned Linear Dimensions, after the first one has been modified, only those subsequent dimension entities having the same dimension extension line direction as the first one will be updated according to the above mentioning. Those not qualified to be updated will be casted out for user interactive operation as usual.

The same rules are also applied to the Ordinate Dimensioning. It is recommended that you should operate on the same kind of dimension entities at a time.

### Procedure:

Follow the procedure below to modify objects

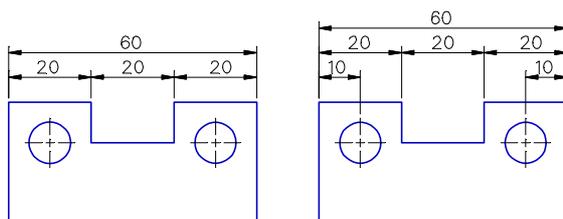
@CMD: **CHANGE** (*return*)

Select Object (+): (*do so*)

LType/Color/Layer/Elevation/Thickness/ALL-same/<change Point>: (*point or option*)

- **Linetype option**  
New Linetype: (*operate on pop-up window*)
- **Color option**  
New Color: (*operate on pop-up window*)
- **Layer option**  
New Layer: (*operate on pop-up window*)
- **Elevation option**  
New Elevation <0.0>: (*value*)
- **Thickness option**  
New Thickness <0.0>: (*value*)

### Example:



Use CHANGE/ALL to 'stretch' out Dimension to make room for new Dimensioning.

# CHDIM

## Change dimensioning Command (TCL)

---

### Purpose:

The **CHDIM** command is used to modify an existing dimensioning by its style, layout, and text annotation in a dynamic dragging interaction manner.

### Description:

The **CHDIM** command lets you modify an existing dimensioning by entering the same dynamic dragging interaction that determines its dimension style, position layout and text annotation, as that provided in its original creation.

You will be asked to pick up the dimension to change by the prompt:

Please select the dimension to change:

The rest operation depends on the type of the dimension entity you have picked up. See the related dimension commands that create various type of dimensioning for further informations on the operation.

### Special Notes:

The **CHDIM** command is an external command provided by the TCL program file "CHDIM.TCL" or "CHDIM.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **CHDIM** command, you may solve the problem by copying the "CHDIM.TCL" to the COMMANDS sub-directory.

### Procedure:

@CMD: **CHDIM** (*return*)

Please select the dimension to change:

...

### Example:

# CHINSERT

## Change Block Instance Command (TCL)

### Purpose:

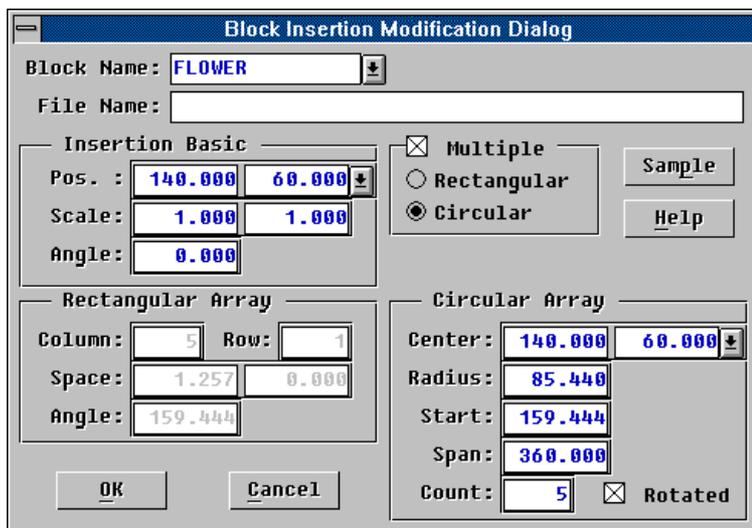
The **CHINSERT** command is used to modify a selected block instance by its insertion parameters via a GUI dialogue operation manner.

### Description:

The **CHINSERT** command lets you modify the insertion parameters of a selected block instance. You will be asked to pick up the block instance to modify by the prompt:

Select the block instance to modify:

And then, **CHINSERT** will pop up a dialogue window, as shown below, about the same as that of **DDINSERT**.



In fact, **CHINSERT** will internally call **DDINSERT** to handle the modification of the block instance entity. See also **DDINSERT** for more details.

### Special Notes:

The **CHINSERT** command is an external command provided by the TCL program file "CHINSERT.TCL" or "CHINSERT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **CHINSERT** command, you may resolve the problem by copying the "CHINSERT.TCL" to the COMMANDS sub-directory.

### Procedure:

@CMD: **CHINSERT** (return)

Select the block instance to modify: (Do so)

...[Dialogue Operation]

### Example:

# CHPROP

## Change Entity Properties Command (TCL)

### Purpose:

The **CHPROP** command is used to modify the properties of selected entities via a GUI dialog window operation.

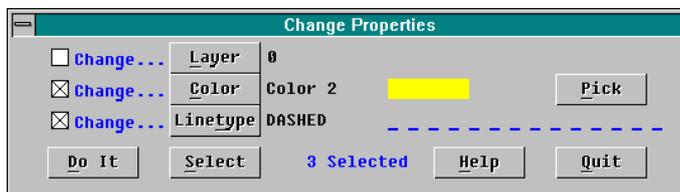
### Description:

The **CHPROP** command lets you change the properties of selected entities via a GUI dialog window operation. By checking on the corresponding check-boxes, you may selectively specify to change only one of the entity properties, such as Layer, Linetype and Color, or all of them of these selected entities. You may also specify new entity properties to change to by reading them from an existing entity, other than specifying each of them individually.

When you enter **CHPROP** command, you will be asked first to select those drawing entities to be modified by their properties via the Object Selection Operation, as it prompts:

Select Objects (+):

After your completion of the object selection, a dialog window as shown in the figure below will pop up.



The operation is simple. You need to specify the desired properties to change to and make a check-mark on the corresponding check-boxes to enable their modifications over the selected entities. And finally, you press the **[Do It]** button to proceed the modification, or the **[Quit]** button to quit the operation.

There are three check-boxes on the left of the dialog window. From top to bottom, each of them indicates the Layer, Color and Linetype to change respectively. After each check-box, there is a button with the corresponding property name as its label. You may press it to specify the new property to change to. That is, press the **[Layer]** button to select the new layer, the **[Color]** button to select the new color, and the **[Linetype]** button to select the new linetype. Each new property being specified will be displayed on the same line after its corresponding button. Once a new property is specified, its corresponding check-box will be automatically checked.

Initially, these check-boxes will be in an unchecked state, and each of the new property will be set to be the same as those of the selected entities, provided that they are all the same.

You may press the **[Pick]** button, on the right of the dialog window, to specify to set the new entity properties by reading from an existing entity. **CHPROP** will close the dialog window temporary and prompt

Pick properties from entity:

asking you to pick up the object in desire. **CHPROP** will read its entity properties and set them as the current new properties to change. Each new property will be checked to see if it will

change the corresponding property of these selected entities. If it will, the corresponding check-box will be checked automatically.

You may press the [**Select**] button to discard the current selection list and do the object selection again. The dialog window will be closed temporary for the object selection. If you are going to modify on the current selection list, enter the "Previous" option to select them again at object selection.

To quit the operation, press the <ESC> key, the right mouse button or the [**Quit**] button from the dialog window. To proceed the entity modification, press the [**Do it**] button.

**Procedure:**

@CMD: **CHPROP** (*return*)  
Select Objects (+): (*do so*)  
... [Dialog Operation] ...

**Example:**

# CHTEXT

## Dialogue based Text Modification Command (TCL)

---

### Purpose:

The **CHTEXT** command is used to modify a single text or a group of selected texts via a GUI dialogue operation manner.

### Description:

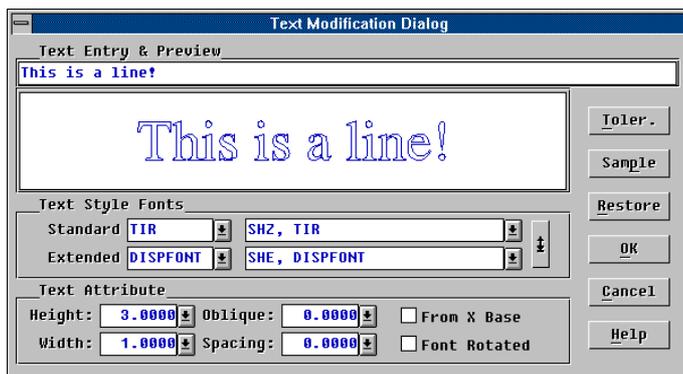
The **CHTEXT** command lets you modify the text content, style and attributes of a single text, or selectively modify the text style and attributes of a group of selected texts, via a GUI dialogue operation manner. It provides two different modes of operations; namely, they are:

- Single Text Modification -- You can modify everything of the selected text by its text content, style font usage and attributes like the text height, width, oblique feature and font orientation.
- Multiple Texts Modification -- If multiple texts are selected, you can selectively change their the text styles or text attributes to the same new values, excluding their text contents.

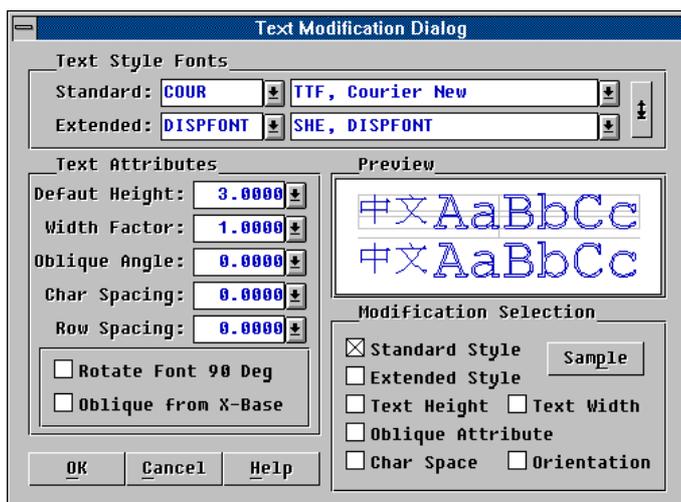
When you enter the **CHTEXT** command, it will ask you to select the texts to modify first via the General Object Selection operation, with the prompt:

Select Text to modify (+):

After the completion of the text selection, **CHTEXT** will pop up a dialog window to proceed the text modification operation, however, depending on the number of text entities being selected. If you have just selected on single text entity, you are going to modify the text in the single mode and **CHTEXT** will pop up the dialogue window as shown below:



If you have selected more than one text entities, **CHTEXT** will pop up the dialogue window as shown below:



The following describes the details of these screen items from the above two dialogue windows.

## Text Entry and Preview

In single text modification mode, you are allowed to modify the text content by clicking the text entry field. The resulted image of the text using current text style will be displayed in the preview window below the text entry field.

The preview window is useful not only in showing the effect of the current text style used for the text creation, but also in justifying the use of the built-in feature text functions and symbols in the text. See the **TEXT** commands for a list of these functions and symbols.

Note that if you erase all the text from the text entry field and make it empty, a message string like "Click Here to Enter Text" will appear in the field. However, it will disappear when you click the field to enter new text again.

You may cut all the text from the entry field to the clipboard by pressing <Ctrl/X> key and paste in the text from the clipboard by pressing <Ctrl/V> key. To enter these two control codes to the entry field, you have to press down the Scroll Lock key. An alternative way to enter control codes is to use the ALT-nnn sequence. The <Ctrl/X> can be entered by ALT-24 and <Ctrl/V>, ALT-22.

## The [Toler.] Button

In single text modification mode, if the text to modify is a tolerance string or you would like to change it into a string of tolerance expression, you may click the [Toler.] button to modify it in tolerance setting method. **CHTEXT** will internally invoke the text input handler **TOLERSTR**, the one called by the **TOLERANCE** command, to handle the tolerance string input, which will help you constructing the tolerance expression using feature text functions.

## The [Sample] Button

In both modes, you may click this button to sample an existing text entity from the drawing to setup the target text style and text attributes. You will be asked to pick up the text by the prompt:

Please select the text to copy style:

Note that in single mode, if the text entry field is empty, the text read from the sample text entity will also be duplicated into the entry field for your further editing.

## Text Style Fonts

There are two parameter items in the "Text Style Fonts" group; namely, they are

- |                 |   |
|-----------------|---|
| <b>Standard</b> | This item specifies the main text style name used for the text in ASCII codes.  |
| <b>Extended</b> | This item specifies the extension text style name used for the text in non-ascii codes, such as the two-byte coded Chinese, Japanese and Korean, and multi-byte coded Thai. |

There are two combo-boxes for each of the above parameter setups. The shorter one on the left shows the name of the text style, which is the name used in the drawing. And, the longer one on the right shows the type and the name of the font, if the font name is available. Both of them provide the same selection set of the available fonts in the system, though in different view of the fonts. Changing either one of them will update the other.

See **DDSTYLE** for detail descriptions about these two same parameter items, including the building of the Loadable Font Table. **CHTEXT** will automatically call **DDSTYLE** to build the Loadable Font Table if the font table is missing or you have clicked at the tiny unnamed button on the right-most side of this "Text Style Fonts" group.

## Text Attributes

The following text attributes are provided in both modes as the target of modification. Note that the display name of these attribute items from the both dialogue windows have a little difference.

- |                      |   |
|----------------------|---|
| <b>Text Height</b>   | This item specifies the size of the text font in height. If this value is negative, the text will be mirrored with respect to the base writing line.  |
| <b>Text Width</b>    | This item specifies the width factor of the text font to generate. It is usually 1. If this value is negative, the text will be mirrored to the opposite of the writing direction.  |
| <b>Oblique Angle</b> | This item specifies the oblique angle of the text font to generate, in units of degrees. It is usually 0. Positive value slant the font backward, and negative value slant it forward. However, if the attribute to "Oblique from X-Base" is enabled, positive value will slant the font up-ward, and negative value, downward. |
| <b>Char Spacing</b>  | This item specifies the character space adjustment in text font generation. A positive value specifies absolute space (additional to the original one defined by the font), negative value specifies relative space ratio over the text height. A zero value will disable this adjustment function.                             |

For multiple text modification, the above parameter items are provided in combo-box. This is because they are provided by the **DDSTYLE** dialogue operation. You may pick such item by its dropdown button to select a value directly from those of common use. For single text modification, these items are provided by pure value entry. You may pick at the value box and enter the value directly.

Note that the item "Row Spacing" from the multiple text modification dialogue window is not effective to the existing text modification.

Besides the above value control attributes, there are also on-off control attributes provided in check boxes:

**Rotate Font 90 Deg** This on-off state selection specifies whether to rotate the text font by 90° Counter-ClockWise or not. This attribute is useful in writing Chinese/Japanese/Korean texts or the likes in vertical manner.

**Oblique from X-Base** This on-off state selection specifies whether to oblique the font from the its base line or not, if the oblique angle is not zero. If it is off, the text font will be oblique in the usual way. However, if it is on, the font will be oblique from its base line, suitable for text font in vertical writing.

## Modification Selection

In multiple text modification, you may specify to update only specific text attributes of these selected text. You may select the text attributes to update by making a check on its corresponding check box from the Modification Selection group.

## Other Buttons

You may press the **[OK]** button to conclude the text modification, press the **[Help]** button to see the helping text, and press the **[Cancel]** button to cancel the **CHTEXT** operation. Pressing <ESC> key also quits **CHTEXT**.

## Procedure:

@CMD: **CHTEXT** (*return*)

Select Text to modify (+):

...[Dialogue Operation]

## Example:

# CIRCLE

## Create Circle Entity Command

---

### Purpose:

The **CIRCLE** command is used to draw a circle.

### Description:

The **CIRCLE** command is a basic drawing command, providing quite a lot of methods for you to create a circle entity. The methods provided here can be categorized into the following:

- **Center and Radius/Diameter**

This is the simplest way to define a circle by directly indicating the center position and its radius value or its diameter value. Additional features in finding solution circle with given center on a specific line or circle are also supported. See later paragraphs.

- **Two points on diameter (2P)**

You will be asked to designate two end points on the diameter of the circle. The center will be at the middle of the two given points, and the radius, the distance from the center point to either one of the given points.

- **Two points on circle with a given radius (2PR)**

You will be asked to designate two points on the circle, and then enter the radius value of the circle required. When the first two points are designated, dragging of the circle of possible solutions will start, and you drag the circle to choose the desired one, and then enter the radius value. The dragged circle will serve as an indication to the solution in desire, since there may be more than one circle which meet the requirement (passing through two point specifications with a given radius).

- **Three points on circle (3P)**

You will be asked to designate three points on the circle. When the first two points are designated, dragging of the circle of possible solutions (passing the cursor point) will start, and you drag the circle to form the desired one, and finally designate the third point. The dragged circle will serve as an indication to the solution in desire, since there may be more than one circle which meet the requirement (passing through three point specifications).

### Point Specification

When you are asked to designate a point, you can designate the point by

- *An absolute point position*, such as the **END**point of an arc segment, **MID**dle point of a line segment, **CEN**ter point of a circle, or direct coordinates input, and so on.
- A **TAN**gent point to a specific object.
- A **PER**pendicular requirement (*normal*) to a specific object.

So with **2PR**, you can find a circle tangent to a line and an arc with a given radius, by picking the line and the arc with the **TAN**-snap function in designating the two points. This is the case that ACAD calls **TTR** function.

However, **TwinCAD** does more than that. You can specify the points for **2PR** or **3P** in any combination of these specifications. You may specify the circle to **TAN** to another circle, **PER** to yet another circle, and **PER** to a line!

Further more, if you have already in mind how the circle is going to be, say **TAN** to this one, **PER** to that one and **PER** to still another one, you can directly enter the shorthand option as "**TPP**" to the first command prompt. **TwinCAD** will issue the proper snap function automatically for you at each prompt for point specification input.

### Null Return Operation and the Last Point

When you issue the **CIRCLE** command, you will be asked to designate the center point of the circle. If you reply to the prompt with a null return, the Last Point will be taken as the center point designation. As the circle command will automatically update the Last Point by the center point of the latest created circle, this default operation will allow you to draw multiple concentric circles easily.

### Default Radius in Center/Radius Creation

The radius of the last created circle will be saved as the default radius for the circle in Center/Radius creation. So, after you have designated the center point, you may simply press the space bar to accept the default radius. This helps you draw multiple circles of same radius.

The default radius value may also serve as a reference value when you specify the radius value in relative mode ("**@val**"). For example, to draw the cross section of a pipe, given an outer diameter and a thickness value, after drawing the outer circle, you may press space bar twice (one for repeating the **CIRCLE** command again, and one for the default center), and then enter the thickness value by "**@-tv**", where tv is the thickness value. The negative sign means the default radius will be decremented.

The only limitation about this default radius operation is that you can not designate the center point by null return, and then reply with null return again for the default radius. The **CIRCLE** command will exit automatically in this case to avoid creating redundant circles. If redundant circles are meant to produce, you may produce it explicitly.

### Sub-command Options

After issuing the **CIRCLE** command, you may enter the following sub-command options to choose the way you want to define a circle:

- 2P** Request to designate 2 points on the circle and on its diameter.
- 3P** Request to designate 3 points on the circle. This includes the 2PR option because at the last point specification, **TwinCAD** also accepts a radius value for the circle.
- TTT** Request to designate 3 points on the circle, using **TAN**, **TAN**, **TAN** snap functions for each point designation respectively.
- TTP** Request to designate 3 points on the circle, using **TAN**, **TAN**, **PER** snap functions for each point designation respectively.
- TPT** Request to designate 3 points on the circle, using **TAN**, **PER**, **TAN** snap functions for each point designation respectively.
- TPP** Request to designate 3 points on the circle, using **TAN**, **PER**, **PER** snap functions for each point designation respectively.
- PTT** Request to designate 3 points on the circle, using **PER**, **TAN**, **TAN** snap functions for each point designation respectively.
- PTP** Request to designate 3 points on the circle, using **PER**, **TAN**, **PER** snap functions for each point designation respectively.
- PPT** Request to designate 3 points on the circle, using **PER**, **PER**, **TAN** snap functions for each point designation respectively.

- PPP** Request to designate 3 points on the circle, using **PER**, **PER**, **PER** snap functions for each point designation respectively.
- TT** Request to designate 3 points on the circle, using **TAN**, **TAN** snap functions for the first two point designation respectively.
- TP** Request to designate 3 points on the circle, using **TAN**, **PER** snap functions for the first two point designation respectively.
- PT** Request to designate 3 points on the circle, using **PER**, **TAN** snap functions for the first two point designation respectively.
- PP** Request to designate 3 points on the circle, using **PER**, **PER** snap functions for the first two point designation respectively.

### Rules to Select the Solutions

As there are always more than one solutions to the circle in the **3P** and **2PR** request, **TwinCAD** takes the following rules to determine the one in desire:

- **Rule 1:** The final circle solution is determined by the dragging circle. A dragging circle is constructed with the cursor position as the third point specification and will change as the cursor moves. When there are more than one qualified solutions, the current dragging circle will become the final solution.
- **Rule 2:** While the dragging circle needs the cursor position as the last condition for determining the solution circle, you may narrow down the selection range by placing the picked points at proper positions on the existing objects such that all undesired dragging circles will be screened out. For example, to draw a circle tangent to a known circle, the potential solution circle will never be realized if the picked point is 90° away from the intended tangent point to the existing circle.
- **Rule 3:** After Rule 2 is applied, if there are still more than one qualified dragging circles, **TwinCAD** will pick up one at random as the final solution. Note that the change of the cursor position may affect the randomly determined result.

Note that if there is no solution for the circle passing through the cursor point, then there will be no circle in dragging. The dragging should not be turned off for circle creation.

### Finding Circles with Center on Specific Object

Additional features to find the solution circle whose center is located on a specific line/arc/circle are also implemented. To do so, you must designate the center point coordinate by using the **PER** snap directive to snap on a specific entity to specify that the center of the solution circle will be on that entity, to the following prompt:

@CMD: CIRCLE 3P/2P/<Center>: **PER** to (select an entity)

Dragging of solution circles are also fully supported and several possible solution modes are provided as listed below: (where **CP** stands for Center point by **PER** to entity)

- CP/<pnt>** Specifying a given point on the solution circle. The center will be determined by the intersection of the snapped entity with the line passing through the point and normal to the snapped entity.
- CP/<rad>/<pnt>** Specifying a fixed radius of solution circle, and then a data point the circle passing through. You may change the radius of circle as many time as you want before designating the last point. Once a radius value is given (explicitly or implicitly by space bar), the dragged circle will be in fixed radius sliding along the snapped entity.

**CP/2P/<pnt1><pnt2>** Specifying to supply additional two points on the solution circle by the sub-command option "**2P**", and then designating the two points.

Note that you may also designate the data points by the **TAN/PER** snap directives to specify the solution circle to be tangent/perpendicular to the specific entities, while its center is located on the snapped entity.

The **PER** to a line in the **CIRCLE/3P** will also generate a circle whose center is on the line.

## Special Notes on Geometry Solutions to Circle

This section is given here only for those who are interested to know the actual number of distinct geometry solutions to circle that **TwinCAD** has provided.

The total number of geometry solutions provided by **CIRCLE** command in combination is **121**. Discounting the 12 trivial cases, and 20 redundant cases (due to the fact that **PER**-to-line in **3P** also means center-on-the-line), there are effectively **89** ways to find a circle, excluding the interaction with the **ELLIPSE** entities.

To give a list of these combination of geometry solutions, notations must be used. The notation for geometry conditions are given below:

<b>X</b>	Passing through a given point.
<b>Ts</b>	Tangent to a given line.
<b>Tc</b>	Tangent to a given circle.
<b>Ps</b>	Perpendicular to a given line.
<b>Pc</b>	Perpendicular to a given circle.
<b>R</b>	With a given radius/diameter.
<b>C</b>	With a given center point.
<b>Cs</b>	Center on a given line.
<b>Cc</b>	Center on a given circle.
<b>?</b>	Any of the point designation <b>{X,Ts,Tc,Ps,Pc}</b> .
<b>??</b>	Any combination of two codes taken from <b>{X,Ts,Tc,Ps,Pc}</b> . There are total 15 effective combinations (e.g. TsTc is treated same as TcTs): XX, XTs, XTc, XPs, XPc, TsTs, TsTc, TsPs, TsPc, TcTc, TcPs, TcPc, PsPs, PsPc and PcPc.
<b>???</b>	Any combination of three codes taken from <b>{X,Ts,Tc,Ps,Pc}</b> . There are total 35 effective combinations (out of 125 possible combinations with different permutations): XXX, XXTs, XXTc, XXPc, XPsPs, XPsPc, XPcPc, TsTsTs, TsTsTc, TsTsPs, TsTsPc, TsTcTc, TsTcPs, TsTcPc, TsPsPs, TsPsPc, TsPcPc, TcTcTc, TcTcPs, TcTcPc, TcPsPs, TcPsPc, TcPcPc, PsPsPs, PsPsPc, PsPcPc and PcPcPc.

Also, the notation for operation sequence are given below:

<b>&lt;...&gt;</b>	User inputs a value or a point designation.
<b>TAN</b>	User inputs the TAN object snap directive.
<b>PER</b>	User inputs the PER object snap directive.
<b>[L]</b>	User picks up a Line.
<b>[C]</b>	User picks up a Circle/Arc.
<b>/</b>	Sequence Delimiter.
<b>{... ...}</b>	Two or more alternative sequences.
<b>2P</b>	Option keyword 2P.
<b>3P</b>	Option keyword 3P.
<b>&lt;pnt&gt;</b>	User inputs a point designation, which may be a specific point coordinate, TAN[L], TAN[C], PER[L] or PER[C].

The following table lists the circle geometry solutions provided by the **CIRCLE** command. Each particular solution is listed with the geometry condition codes and its operation sequence.

- *Given a center coordinate*  
**CR**            <Center>/<Radius>  
**C?**            <Center>/<pnt>
- *Center on a given line with a diametric point*  
**Cs?**            PER[L]/<pnt>
- *Center on a given line with given radius and passing a point*  
**CsR?**          PER[L]/{<Radius>/<pnt>|2P/<pnt>/<Radius>}
- *Center on a given line and passing two given points*  
**Cs??**          PER[L]/2P/<pnt>/<pnt>
- *Center on a given circle with a diametric point*  
**Cc?**            PER[C]/<pnt>
- *Center on a given circle with given radius and passing a point*  
**CcR?**          PER[C]/{<Radius>/<pnt>|2P/<pnt>/<Radius>}
- *Center on a given circle and passing two given points*  
**Cc??**          PER[C]/2P/<pnt>/<pnt>
- *Given two points on diameters (defining diameter line)*  
**??**            2P/<pnt>/<pnt>
- *Passing two points with given radius*  
**??R**          3P/<pnt>/<pnt>/<Radius>
- *Passing three points*  
**???**          3P/<pnt>/<pnt>/<pnt>

The following 12 geometry conditions are trivial cases:

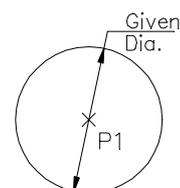
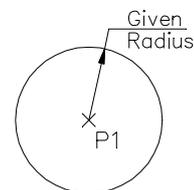
**CPs, CsTs, CsPs, CsPsPs, CcPs, CcPsPs, XTs, TsTs, TsTc, TsPs, TsPc** and **PsPsPs**.

Examples to read the geometry condition codes:

- PsPsPs** Find a circle perpendicular to three given lines.  
**XTsTc** Find a circle passing a given point, tangent to a given line and a given circle.  
**CcXTs** Find a circle of which the center is on a given circle, passing a given point and tangent to a given line.

### Procedure:

- **To find a circle by center and radius**  
 @CMD: **CIRCLE** (return)  
 3P/2P/<Center point>: (point)  
 Diameter/<Radius:(val)>: (value)
- **To find a circle by center and diameter**  
 @CMD: **CIRCLE** (return)  
 3P/2P/<Center point>: (point)  
 Diameter/<Radius:(val)>: **D**  
 Diameter: (value)



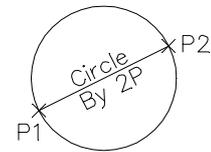
- **To find a circle by 2 points on its diameter**

@CMD: **CIRCLE** (return)

3P/2P/<Center point>: **2P**

First point on diameter: (point)

Second point on diameter: (point)



- **To find a circle by 3 points on it**

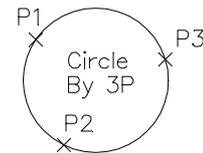
@CMD: **CIRCLE** (return)

3P/2P/<Center point>: **3P**

First point of circle: (point)

Second point of circle: (point)

Third point of circle/<Radius>: (point)



- **To find a circle by 2 points on it with a given radius**

@CMD: **CIRCLE** (return)

3P/2P/<Center point>: **3P**

First point of circle: (point)

Second point of circle: (point)

Third point of circle/<Radius>: (value)

- **To find a circle by TTT options** (same as those for other options)

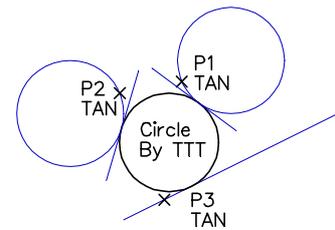
@CMD: **CIRCLE** (return)

3P/2P/<Center point>: **TTT**

First point of circle: TAN to (pick object)

Second point of circle: TAN to (pick object)

Third point of circle/<Radius>: TAN to (pick object)



- **To find a circle whose center is on an entity, by another two points**

@CMD: **CIRCLE** (return)

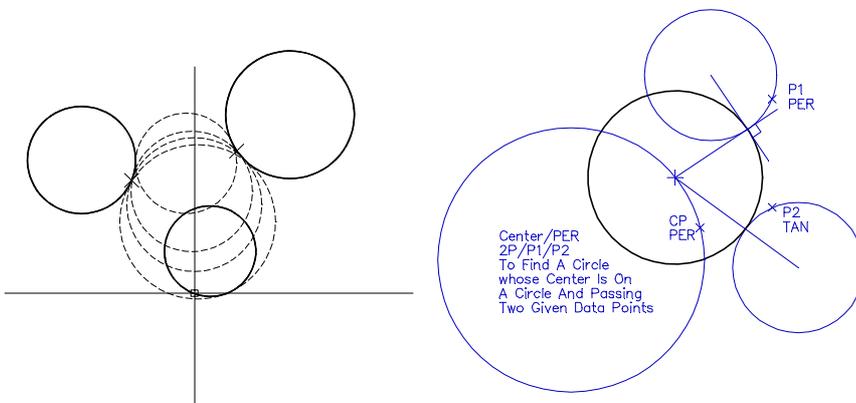
3P/2P/<Center point>: **PER** to (pick object)

Diameter/<Radius:(val)>: **2P**

First point of circle: (point)

Diameter/<Radius:(val)>: (point)

**Example:**



Example Showing the Dragging Circle

# CLEANUP

## Clean Up Garbage and Refresh System Resource

---

### Purpose:

The **CLEANUP** command is used to refresh the system resource and collect memory garbage so as to make it available again.

### Description:

The **CLEANUP** command lets you reconstruct the current drawing database and refresh the system resource.

The **CLEANUP** command will save the current drawing to a temporary file, clear up every things, and reload the drawing file back again. This will force **TwinCAD** to rebuild the internal working structure of the drawing. The system resource taken up by the **UNDO** buffer, Operation History Record, and the garbage produced by drawing editing, are released to the system again.

Note that the unreferenced anonymous block definition will be removed automatically by the **CLEANUP** command.

### Special Notes

The **CLEANUP** command will be issued automatically whenever the drawing space is almost full (of gabages maybe, after long period of operations). **TwinCAD** will prompt:

Drawing space is almost full -- press any key to clean up gabages.

If the almost-full condition can not be released, **TwinCAD** will then automatically save the drawing to disk (in the same way done by the AutoSave feature) and then issue the following message:

Your drawing is too large and is about to exceed the capacity of your version of TwinCAD. Please try to purge off some-unwanted blocks.

This message will be given only once after the condition is met and persisted.

### Procedure:

Enter the **CLEANUP** command to **TwinCAD** command prompt:

@CMD: **CLEANUP**

and after a while, **TwinCAD** will report the completion of the command.

# CLEARALL

## Clean Up Everything for A New Drawing File

---

### Purpose:

The **CLEARALL** command is used to clean up everything in the system resource for a brand new empty drawing file.

### Description:

The **CLEARALL** command lets you create a brand new empty drawing file by cleaning up everything in the system resource. You will be asked for confirmation before the operation proceeds.

This command is usually used to build a user prototype drawing file from the very beginning. The empty drawing file will contain only Layer "0", "CONTINUOUS" linetype and system variables.

### Procedure:

Enter the **CLEARALL** command to **TwinCAD** command prompt:

```
@CMD: CLEARALL
Clear all the entities ? <N>
New File:
@CMD:
```

**TwinCAD** reports a brand new empty drawing file in creation.

# CLOAD

## Load Commands from TCL Programs

---

### Purpose:

The **CLOAD** command is used to load in TCL programs.

### Description:

The **CLOAD** command lets you load in specific TCL programs from external disk storage. All the **COMMANDS** and **FUNCTIONS** defined in the TCL programs will become a part of **TwinCAD**.

**TCLtm**, short name for **TwinCAD Command Language<sup>tm</sup>**, is a built-in C-like programming language in **TwinCADtm** package. It provides the users and application developers a way to write their own programs and add commands and functions to **TwinCAD**, with a very powerful high-level language.

**TCL** mimics the standard C language, with several additional features to make the programming much easier for CAD/CAM applications. Such features include predefined geometry data structures, system variables, TCL geometry registers, and the related operations in expressions.

**TCL** is designed to have a very close relationship with **TwinCAD**'s command line input. User may directly enter a TCL expression to the command line whenever a value, a point, or a string is required, as long as the expression can be evaluated to return the proper type of value. User may load in commands or functions written in TCL from a disk file, and enter the command or reference the functions in expressions as if it was originally built in **TwinCAD**. There is no limits on the size, memory storage, and the number of TCL programs that can be loaded into **TwinCAD**.

See ***TCL Command Language Reference Manual*** for detail information about the TCL programming.

### Procedure:

Enter the **CLOAD** command to **TwinCAD** command prompt:

```
@CMD: CLOAD
```

and **TwinCAD** will pop up the file window for you to select the TCL program file to load. After loading the program file, **CLOAD** will report the names of the extended commands and functions loaded from the file.

<b>'CMASK</b>
---------------

## **Setup Color Mask for Object Selection Command (TCL)**

---

### **Purpose:**

The **CMASK** command is used to setup the color mask for the subsequent object selection.

### **Description:**

The **CMASK** command lets you setup the color mask for the subsequent object selection via a GUI dialog manner. It will open a dialog window containing 16 check-boxes for the 16 pen colors. You may place a check-mark on a pen color to specify that the objects with the corresponding pen color will be allowed to select during the object selection operation when the selection masking is enabled. See **SELECT** command for details about the object selection operation.

### **Special Notes**

The color mask is controlled by the system variable **COLORMASK**, which is an integer bit-flag variable with volatile attribute and will be reset to 0 each time **TwinCAD** is initialized. The 16 color numbers corresponds to the 16 bits of the integer from bit 0 (LSB) to bit 15 (MSB). If a bit is set to 1, the entity with the corresponding color will be rejected by the object selection operation, provided that the selection mask is enabled. Color of **[ByLayer]** and **[ByBlock]** are resolved to the actual color used for the entity before the masking.

### **Procedure:**

```
@CMD: COPY (return)  
Select Objects (+): 'CMASK  
...[Dialog Operation to setup color mask]...  
Select Objects (+):  
...
```

**'CMDLIST**

## List All Registered Commands and Function

---

**Purpose:**

The **CMDLIST** command is used to list all registered commands and function in **TwinCAD**.

**Description:**

The **CMDLIST** command lets you list out all the commands and functions known to **TwinCAD**. It can be used to identify whether specific commands or functions are supported or not, especially those defined by the TCL programming, since they must be loaded before use. It can be used as a simple help to the operator in case the name of a specific function or command is forgotten.

The listing generated by **CMDLIST** consists of three parts:

- **TwinCAD's** Built-in Commands
- **TCL** Runtime Function
- Extended Commands and Function

However, not all the listings need to be generated each time **CMDLIST** is issued. Most of the time, you may want to issue **CMDLIST** just to see the names of the loaded extended commands and function. So, when you issue **CMDLIST** command, you may choose to generate only the listing of the loaded extended commands and function by replying to the prompt with a null return.

You may enter the sub-command options "**F**" to generate the listing of all TCL runtime functions, and enter "**C**" to generate the listing of **TwinCAD's** built-in commands. Or, you may enter "**ALL**" to generate all the listings.

**Special Notes**

This command is very useful in generating the 'Feature List' for an OEM version. Some OEM versions of **TwinCAD** does not support all the functionality that this document has revealed.

**Procedure:**

To list all the supported commands and functions, issue the **CMDLIST** to **TwinCAD** command prompt:

```
@CMD: CMDLIST -- ALL/Command/Function/<External Loaded Only>: ALL  
(Listing generated in the command area)
```

# 'COLOR

## Set New Entity Color Command

---

### Purpose:

The **COLOR** command is used to specify the color number for new entities of subsequent creation.

### Description:

The **COLOR** command lets you specify the default color for the subsequent new entity creations. This is an ACAD-compatible command.

As the assignment of color to each color number depends on the contents of the system initial file, it is better to use the **ECOLOR** command instead. Because the **ECOLOR** command pops up a slide window for color selection, it will be much easier for you to set the desired color. The valid range for color number is from -2 to 15. The -1 is used to indicate the color by layer, or **[BYLAYER]**, and -2, the color **[BYBLOCK]**. And, the value from 0 to 15, specify the 16 colors.

Note that if the color number is not **[BYLAYER]**, it will not be changed when you switch the current layer. Therefore, it is recommended to set the color as **[BYLAYER]** and let the layer control the display color.

### Procedure:

Enter the **COLOR** command to **TwinCAD** command prompt:

```
@CMD: COLOR
```

and **TwinCAD** will prompt

```
New entity color:
```

for you the enter the desired color number.

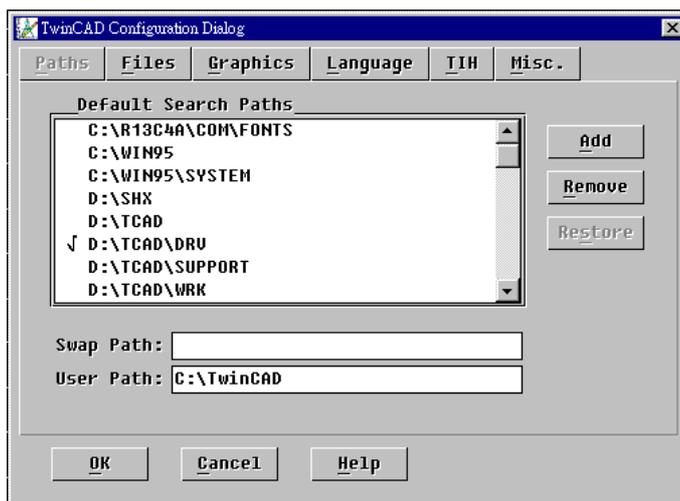
### Example:

**'CONFIG****Configuration Setup Command (TCL, Win)****Purpose:**

The **CONFIG** command is used to setup general system configuration via a GUI dialogue operation.

**Description:**

The **CONFIG** command lets you setup the general system configuration via a GUI dialogue window operation. It provides a property-sheet like user interface and is divided into several pages of property-sheet. Each of them can be accessed directly by picking at the corresponding sheet title on the top of the dialog window. The following sections will describe them in details.

**Paths**

This page setups the following items:

**Default Search Paths** The default search paths are the paths that must be searched through for a missing file automatically. Whenever **TwinCAD** fails to open a given file for reading, it will search through these paths for it. Note that the Windows directory, the Windows system directory, the directory where **TwinCAD** resides and all its sub-directories are always default search paths, regardless of being explicitly specified or not.

You may add new path to the list by pressing the **[Add]** button and choose the directory path from the pop-up file window. You may remove existing paths from the list by selecting them first (pick to select and to un-select) and then pressing the **[Remove]** button. A selected path will have a check mark before it. You may press the **[Restore]** button to restore the change of the path list.

This item corresponds to the **TCADPATH** entry from the **[General Setup]** section in the system initial file.

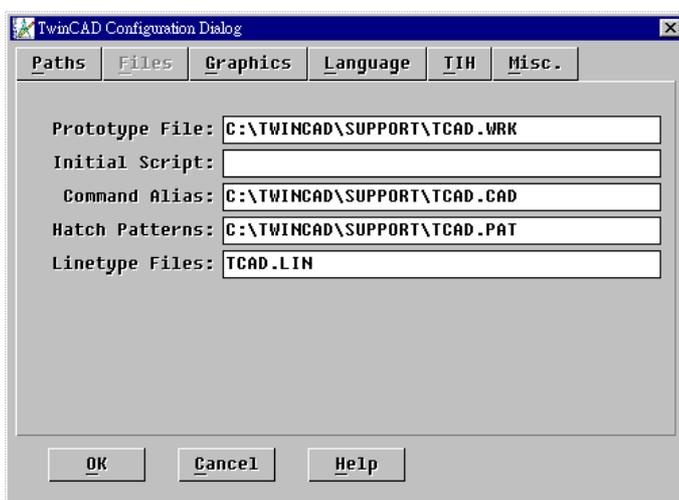
**Swap Path** The Swap Path is the path where **TwinCAD**'s virtual memory manager used to swap memory data. The default will be the root directory of the temporary drive returned by the Windows, which is usually the the first hard drive in the system.

This item corresponds to the **MainSwapPath** entry from the **[General Setup]** section in the system initial file.

**User Path** The User Path is the path where **TwinCAD** may use to save and to retrieve specific user data files (such as these SET files of individual commands). If this path is not given, it defaults to the current directory, which may be changing and transparent to the user.

This item corresponds to the **UserWorkingPath** entry from the **[General Setup]** section in the system initial file.

## Files



This page setups the following items:

**Prototype File** This item specifies the default prototype drawing file that **TwinCAD** will load in automatically in creating a new empty drawing. The internal default is "TCAD.WRK", if it is not specified.

This item corresponds to the **PrototypeWrkFile** entry from the **[General Setup]** section in the system initial file.

**Initial Script** This item specifies the script file to execute automatically after **TwinCAD** is started. There is no default for this item. It corresponds to the **InitialScriptFile** entry from the **[General Setup]** section in the system initial file.

**Command Alias** This item specifies the command alias file to load in after **TwinCAD** is started. The command alias file is used to defined alias commands. The default is "TCAD.CAD", if it is not specified.

This item corresponds to the **CommandAliasFile** entry from the **[General Setup]** section in the system initial file.

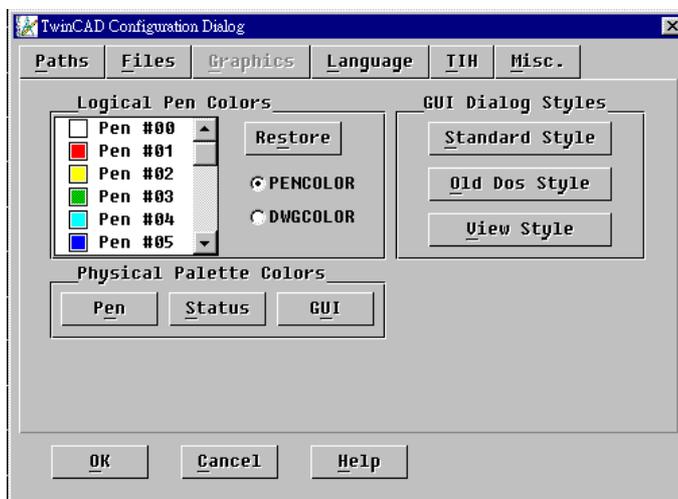
**Hatch Patterns** This item specifies the hatch pattern definition file to load in after **TwinCAD** is started. The hatch pattern definition file is used to stored predefined hatch patterns. The default is "TCAD.PAT", if it is not specified.

This item corresponds to the **HatchPatternFile** entry from the **[General Setup]** section in the system initial file.

**Linetype File** This item specifies the linetype definition file to load in after **TwinCAD** is started. The linetype definition file is used to stored predefined linetypes. The default is "TCAD.LIN", if it is not specified.

This item corresponds to the **ExtLinetypeFile** entry from the **[General Setup]** section in the system initial file.

## Graphics



This page setups graphic related configuration.

### Logical Pen Colors

The logical pen colors group setups the 16 logical drawing pen's color from pen color 0 to pen color 15. There are two set of such drawing pen color assignment:

**PenColor** The native drawing pen color setup for **TwinCAD**. **TwinCAD** does not regulate the actual color used for each pen color number. However, a default assignment has been given since its DOS release. It is recommended now that this should be set to be compatible with the DWG file specification.

**DWGColor** This is the pen color set for DWG file compatibility. **TwinCAD** will automatically switch to use this pen color set whenever a DWG file is loaded.

You can choose to setup either one of the above two sets of logical pen color assignments by selecting on their corresponding radio buttons.

To change a specific logical pen color assignment, pick at the pen from the list box directly. A logical pen color assignment dialog window will pop up for you to choose for the new pen color. You can choose it from the 16 physical pen colors, of which the actual colors are setup separately.

## Physical Palette Colors

There are three buttons in this group.

- Pen** Press this button to setup the physical pen colors used in the drawing. The physical pen color is the palette color used by **TwinCAD** to realize the drawing's logical pen. Currently, only 16 palette colors are supported. **TwinCAD** uses the physical palette color index 0 as the drawing's background in the screen. So, change palette color index 0, will change the drawing's background after drawing refresh or regeneration.
- Status** Press this button to setup the physical palette colors used in drawing status display, such as the object high-light, various kind of cursors and status icons.
- GUI** Press this button to setup the physical palette colors used in the GUI elements created by TCL runtime functions as well as those created by the **TwinCAD** commands that are compatible with its DOS counterpart. A special window class called TCAMGUI is implemented in **TwinCAD** to simulate the TCAM GUI used in its DOS counterpart. The TCAMGUI assumes the display is a standard VGA using 16 color palettes.

A physical palette color setup dialog window will pop up when you press either one of the above two buttons.

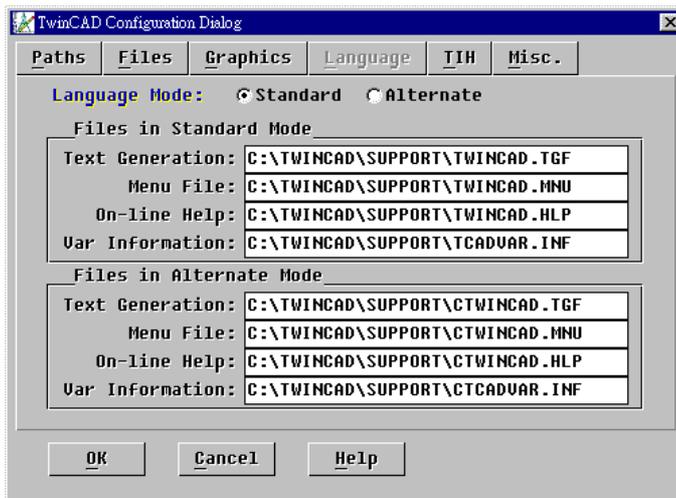
## GUI Dialog Style

There are three buttons in this groups:

- Standard Style** Press this button and all the dialog windows created by **TwinCAD** commands will be in gray background.
- Old Dos Style** Press this button and all the dialog windows created by **TwinCAD** commands will be in colorful background as those in its DOS counterpart.
- View Style** Press this button to see an example of the dialog window.

The size, location and color attributes of each dialog window created by **TwinCAD** command is stored in the system initial file. The above function actually modifies these setups. Note that once the style is changed, it can not be restored by pressing the main [**Cancel**] button.

## Language



Although its DOS version counterpart of **TwinCAD** supports multi-lingual operating environments, **TwinCAD** is basically a bilingual CAD system for Windows in that it supports a **SWITCH** command that can be issued at anytime to switch the operating environment between the two different language modes.

The two language modes supported in **TwinCAD** are

**Standard Mode** English language.

**Alternative Mode** Local language supported by Windows.

There are several language dependent files need to be specified in both modes:

**Text Generation File** System message text generation file.

**Menu File** The current default menu file. This setting will be modified automatically by **MENU** command.

**On-line Help** The help file invoked by **HELP** command.

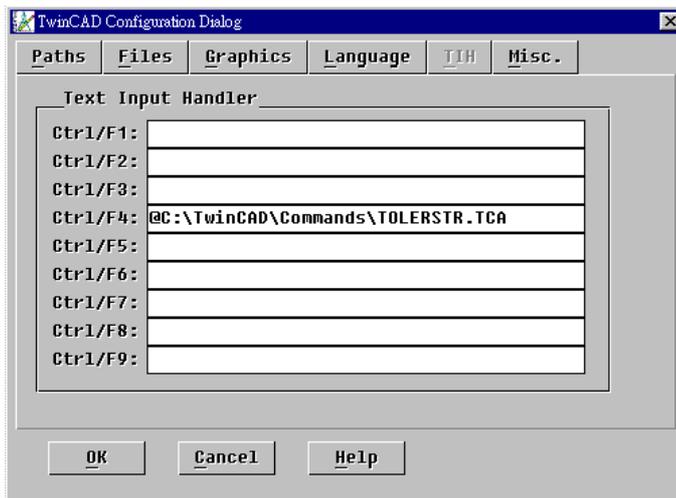
**Var Information** Variable information file. A simple text file containing short description to system variable, used in **SYSVAR**, **DIMVAR** and **STYLE** commands.

For each of the above entry, **TwinCAD** will have a default setting internally according to the Windows language version:

<b>Standard Mode, English</b>	"TWINCAD.TGF", "TWINCAD.MNU", "TWINCAD.HLP", "TWINCAD.VAR".
<b>Traditional Chinese</b>	"CTWINCAD.TGF", "CTWINCAD.MNU", "CTWINCAD.HLP", "CTWINCAD.VAR".
<b>Simplified Chinese</b>	"BTWINCAD.TGF", "BTWINCAD.MNU", "BTWINCAD.HLP", "BTWINCAD.VAR".
<b>Japanese</b>	"JTWINCAD.TGF", "JTWINCAD.MNU", "JTWINCAD.HLP", "JTWINCAD.VAR".
<b>Korean</b>	"KTWINCAD.TGF", "KTWINCAD.MNU", "KTWINCAD.HLP", "KTWINCAD.VAR".

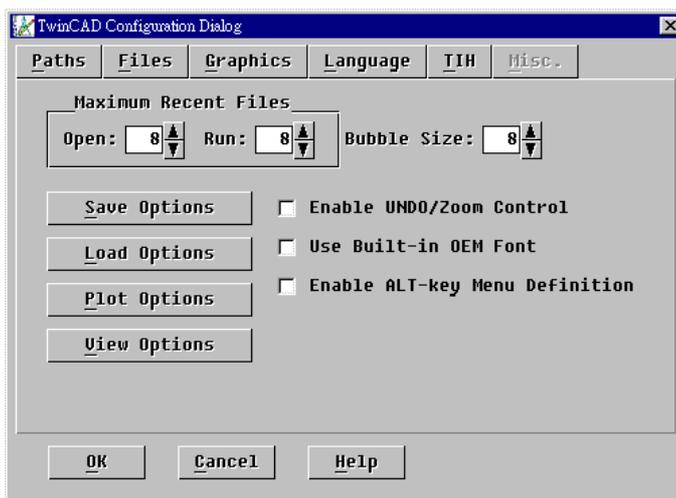
<b>Thai</b>	"TTWINCAD.TGF", "TTWINCAD.MNU", "TTWINCAD.HLP", "TTWINCAD.VAR".
<b>Others</b>	"TWINCAD.TGF", "TWINCAD.MNU", "TWINCAD.HLP", "TWINCAD.VAR".

## TIH (Text Input Handler)



This page setups the text input handler associated with each <Ctrl/Fn> key. Currently available text input handler in TCL program is the "TOLERSTR.TCL" for tolerance expression input. You may assign it to <Ctrl/F4> for DOS version compatibility. See **TEXT** command for further details.

## Misc. (Miscellaneous)



### Maximum Recent Files

TwinCAD currently supports two MRU (Most Recently Used) lists in the menu. You may specify the maximum number of entries allowed for each list from the Maximum Recent Files group:

**Open** This value entry specifies the maximum entries in \$OPEN list, which saves the most recent open drawing files.

**Run** This value entry specifies the maximum entries in \$RUN list, which saves the most recent executed TCL program files.

### Bubble Size

This is a value entry specifying the logical font size used for the bubble text (helping text in a pop-up tiny window with yellow background). If it is 0, then the current Windows system default font and size will be used. Otherwise, the font name will always be "MS Sans Serif".

If the font size is positive, **TwinCAD** will check the Windows language version and specify the local language character set to use for the font. If the font size is negative, it specifies ANSI set. The actual font used will be determined by the Windows Font Mapper.

This item corresponds to the **BubbleFontSize** entry from the **[General Setup]** section in the system initial file.

### Enable Undo/Zoom Control

This is an option check box control. If this option is enabled, the change of the screen drawing view (via **ZOOM/PAN** commands) will be included in the **UNDO/REDO** mechanism. Note that the change of drawing view has its own undo/redo mechanism (via **ZP/ZN** commands).

### Use Built-in OEM Font

This is an option check box control. If this option is enabled, **TwinCAD** will use its internal built-in bitmap font to display IBM-extended Ascii characters. If this option is not enabled, **TwinCAD** will use Windows OEM font for the message text display. Note that **TwinCAD** will always use built-in bitmap font under NT machine, despite of this setting.

If you have problem with the message text display using the IBM-extended ascii character set in **TwinCAD**, you should make a check on this option. Some language version of Windows does not support the IBM-extended ascii character set in its system OEM font.

### Enable Alt-key Menu Definition

This is an option check box control. If this option is enabled, the Alt-Key definition from the menu file will be valid and override the short-cut key in the pop-up menu invocation. Otherwise, the Alt-key definition from the current loaded menu file will be ineffective.

### Miscellaneous Option Setup Buttons

There are several option setup buttons in this page that invoke other independent option setup dialog.

**Save Option** This button invokes the **SAVEOPT** command to setup the drawing saving related options. See **SAVEOPT** command for details.

**Load Option** This button invokes the **LOADOPT** command to setup the drawing loading related options. See **LOADOPT** command for details.

**Plot Option** This button invokes the **PLOTOPT** command to setup the printing/plotting related options. See **PLOTOPT** command for details.

**View Option** This button invokes the **VIEWOPT** command to setup the screen drawing view related options. See **VIEWOPT** command for details.

Note that the setup operation from these above option button can not be canceled by the main **[Cancel]** button.

### Other Buttons

Except noted otherwise, after completing all the setup operation, you will have to press the **[OK]** button to confirm the change and terminate **CONFIG** to make them become effective. Or, you may press the **[Cancel]** to quit **CONFIG** operation. You may also press <ESC> key or the right mouse button to quit the operation.

### Special Notes:

The **CONFIG** command is an external command provided by the TCL program file "CONFIG.TCL" or "CONFIG.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **CONFIG** command, you may solve the problem by copying the "CONFIG.TCL" or "CONFIG.TCA" to the COMMANDS sub-directory.

### Procedure:

Enter the **CONFIG** command to the system command prompt:

```
@CMD: CONFIG
```

```
...[Dialogue Operation]...
```

# COPY

## Duplicate Selected Objects Command

---

### Purpose:

The **COPY** command is used to duplicate selected objects in the drawing.

### Description:

The **COPY** command lets you duplicate selected objects at designated locations in the drawing.

You are asked first to select objects for duplication (see **SELECT** command for details). And then, you are asked to specify the base point (a reference point from these objects) for the duplication. After that, you may drag the objects to anywhere in the drawing, and place down the objects by designating the point desired. The displacement for the new copy of the selected objects is then calculated by subtracting the base reference point from the designated point.

You may directly specify this displacement, using the following methods.

- Input it in absolute format to the first prompt (asking for base point), and reply to the second prompt with a null return (space bar).
- Input it in relative format to the first prompt. **TwinCAD** will directly take it as the displacement value, and the second prompt will not appear in this case.

You can make multiple copies of these objects by entering sub-command option "**M**" to the first prompt for base point. **TwinCAD** will continue to ask you to place down the objects one after another, until you reply to the prompt with a null return, which ends the operation.

If you have selected too many objects, the dragging may be slow. You may turn off the dragging by the **DRAGMODE** command.

See also **ARRAY** command for multiple copying objects in rectangular or circular pattern.

## Copy Entities in 3-D Space

If the system variable **DO3DMODE** is 0 (OFF), which is the default, the **COPY** command will copy the selected entities in 2-D mode. That is to say, the Z-axis offset for each copy is always zero. However, if the system variable **DO3DMODE** is 1 (ON), the **COPY** command will read the Z-component in the point designation, and use it in the copying operation when applicable.

## Special Notes

By default, the properties of each copied entity will be directly copied from its original entity. However, you may instruct the **COPY** command to use the current default properties (those set up by **EMODE** command) for these newly copied entities, by setting the system variable **USE\_EMODE** to 1 (ON). Affected entity properties are layer, color, linetype, elevation and extrusion thickness. Note that if the system variable **DO3DMODE** is also ON, then the elevation and extrusion thickness will not be affected by the setting of **USE\_EMODE**.

Also note that the result of the **COPY** command is affected by the current **ECS** setting from the TCL runtime function **set\_ecs()**, which means you may copy objects from one coordinate plane to another coordinate plane. However, the 3D-ECS/UCS-plane operation is not yet explicitly supported by current version.

## Procedure:

- **To make a single copy of selected objects**
  - @CMD: **COPY** (*return*)
  - Select Objects (+): (*do so*)
  - <Base point or Displacement>/Multiple: (*point*)
  - Second point of displacement: (*point*)
  - or
  - <Base point or Displacement>/Multiple: (*relative displacement*)
  - Second point of displacement: (*space bar*)
- **To make multiple copies of selected objects**
  - @CMD: **COPY** (*return*)
  - Select Objects (+): (*do so*)
  - <Base point or Displacement>/Multiple: **M**
  - Base point: (*point*)
  - Second point of displacement: (*point*)
  - ...
  - Second point of displacement: (*point*)
  - (*Exit by space bar or by Ctrl/C*)

## Example:

# COPYCLIP

## Copy Objects to Clipboard Command (Win)

---

### Purpose:

The **COPYCLIP** command is used to copy selected objects to the Windows Clipboard in various formats.

### Description:

The **COPYCLIP** command lets you copy selected objects to the Windows Clipboard in the following formats:

- |                     |   |
|---------------------|---|
| <b>Text</b>         | If the selected objects contain Text, Attribute Tag and Dimension entities, these text informations can be extracted and saved to the clipboard in Windows TEXT format, provided that the Text option is enabled.   |
| <b>Meta Picture</b> | If the Meta option is enabled, a picture containing solely these selected objects will be created and sent to the clipboard in Windows Meta File format. However, the picture resolution will be limited to 2048 units in either picture width or picture height, whichever is larger.  |
| <b>WRK Drawing</b>  | If the WRK option is enabled (which is the default condition), the selected objects will be saved to the clipboard in TwinCAD's native drawing format, which preserves all the original drawing informations. In fact, when you copy the selected objects to the clipboard in this format, it functions just like that you have issued <b>WBLOCK</b> command to write them out to an external storage. And, when you paste it in from the clipboard using <b>PASTECLIP</b> command, it functions like that you are inserting them back from that external storage using "INSERT *= <i>extfile</i> " command, where <i>extfile</i> stands for external drawing filename. |
| <b>DWG Drawing</b>  | If the DWG option is enabled, the selected objects will be saved to the clipboard in AutoCAD R12 drawing format, which can be directly pasted into AutoCAD R12/R13 drawing.   |

When you enter **COPYCLIP** command, you will be asked first to select those objects to be saved to clipboard via the Object Selection Operation as it prompts:

Select Objects (+):

After the completion of the object selection, you are then asked to specify the saving format by entering the corresponding keyword to the prompt:

Clipboard format -- Wrk/Dwg/Meta/Text/ALL <W>:

where the option keywords are

- |            |  |
|------------|--|
| <b>W</b>   | WRK Drawing file format, which is the default. |
| <b>D</b>   | DWG Drawing file format.                       |
| <b>M</b>   | Windows Meta File picture format.              |
| <b>T</b>   | Windows Text format.                           |
| <b>ALL</b> | All the available formats.                     |

The prompt will be repeated with the current effective options displayed in the pair of parenthesis, until you press space bar to conclude the specification. Note that each time a

valid keyword is entered, it toggles the corresponding option once, except the "ALL" keyword which overrides all the options.

If either the WRK or the DWG drawing file format is specified, **COPYCLIP** will continue to prompt:

Insert Base Point:

for you to specify an insert base point of those selected objects. This location is used in insertion alignment of the drawing when it is pasted into another drawing.

### Special Notes

To copy the current drawing view to Windows Clipboard in Bitmap format, use <Ctrl/K> function key. The <Ctrl/K> function key will also save the image in Meta File format simultaneously. However, if you are going to insert a high quality drawing image into a desktop publisher, it is recommended to use **BMPOUT** or **WMFOUT** or other image export commands to generate the image file. These commands allow you to have more controls about the image size, resolutions and pen widths that affect the quality of the export image.

The use of **COPYCLIP** command is intended to copy a part of a drawing from one drawing file into another drawing file.

### Procedure:

@CMD: **COPYCLIP** (*return*)  
Select Objects (+): (*do so*)  
Clipboard format -- Wrk/Dwg/Meta/Text/ALL <W>:  
...  
Insert Base Point: (*if Wrk/Dwg format specified*)

# CSPLINE

## Cubic Spline Curve Fitting Command

### Purpose:

The **CSPLINE** command is used to fit a given data set of points with a cubic spline curve.

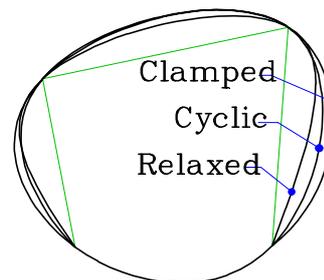
### Description:

The **CSPLINE** command lets you fit a smooth curve over a set of data points via the **Cubic Spline Curve Fitting** technique. The set of data points must be given in the form of a polyline. The curve thus fitted will pass through each vertex of the polyline.

**TwinCAD** actually solves the linear matrix equations by swapping the coefficient data to disk file if the matrix is too large. The solving algorithm had been optimized such that the number of swapping is minimized. Therefore, there is no limits on the number of data points that can be fitted by the same curve. However, for current release, the maximum number of vertexes of a polyline is 65535, which sets a tangible limit.

**TwinCAD** has provided three kinds of boundary conditions for you to apply in fitting the cubic spline curve; namely, they are:

- **CLAMPED** condition -- The tangent direction of the spline at both ends are clamped in specified directions.
- **RELAXED** condition -- There is no clamping effect on both ends of the spline. They are just joined at the ends, and the curvature tends to be 0 at both ends.
- **CYCLIC** condition -- Assuming that the spline curve is closed and continuous at each point of joint, no clamping effect is imposed. A closed polyline will always be fitted over under this **CYCLIC** condition.



Note that **TwinCAD** will use the normalized parameter for each of the spline segments, instead of using each of the chord lengths, in current release of **CSPLINE** command.

When you enter the **CSPLINE** command, you will be asked to select the polyline that defines the curve first. If the polyline is not closed, you will be asked then to indicate the tangent direction of the spline at the start point:

Cyclic/Relaxed/<Start point clamped direction>:

where you may enter the sub-command options:

**C** - Cyclic, specifying **CYCLIC** condition to be used.

**R** - Relaxed, specifying **RELAXED** condition to be used.

If you press space bar in response to the prompt, the boundary condition will be default to the clamped condition with the tangent direction taken from the one tangent to the circle passing through that end point and the next two adjacent data points. The tangent direction given at the start point specifies the tangent vector of the spline leaving the start point.

For clamped condition, **TwinCAD** will continue to prompt

<End point clamped direction>:

asking again for the clamped condition at the end point. Note that the tangent direction given at the end point specifies the tangent vector of the spline entering into the end point.

See the procedure section for further details.

After the curve is generated, **TwinCAD** will ask you whether to erase the original polyline that defines the curve or not. You may choose to keep it, if you still want the polyline to produce another curve in different fineness for you.

## Generation of the Spline Curve

The resulting cubic spline curve is expressed in the form of a polyline consisting of smooth arc segments that approximate the spline, of which the fineness is controlled by the system variable **SPLINESEG** that specifies the additional number of points to locate between two given consecutive data points for the dual-arc-segment approximation to apply to each of these divisions. Note that the inflection point, if there is any on the curve, will be located and treated as a given data point in the dual-arc-segment approximation.

And thus, a value of 0 in the **SPLINESEG** specifies to generate two arc segments between two given data points, and a value of 1, four arc segments, and a value of 2, six arc segments and so on. Should there be an inflection point in between, the number of arc segments will be doubled.

Be aware that the larger value of **SPLINESEG** is, the more arc segments are produced, and then more resources are taken to express the curve. With the dual-arc-segment approximation technique that **TwinCAD** has provided, it is not necessary to have a large number of points interpolated between given data points to have a good result. Usually, a value of 0 to 2 is quite satisfactory, depending on the requirement of the application.

If the fineness control is important to your application, you may repeat the command operation over the same data set (polyline) for several different **SPLINESEG** values. Choose the optimal one by comparing their difference. Use **UNDO** command to undo all the test curves (so that the used system resource can be made available again), and then issue the **CSPLINE** command to obtain the optimal curve with the appropriate **SPLINESEG** value.

### Procedure:

- **To fit cubic spline curve on CYCLIC condition**  
 @CMD: **CSPLINE** (*return*)  
 Select polyline: (*pick one*)  
 Cyclic/Relaxed/<Start point clamped direction>: **C**  
 Done! -- New polyline with *nnn* entity segments created.  
 Delete old polyline? (*Y or N*)
- **To fit cubic spline curve on RELAXED condition**  
 @CMD: **CSPLINE** (*return*)  
 Select polyline: (*pick an open one*)  
 Cyclic/Relaxed/<Start point clamped direction>: **R**  
 Done! -- New polyline with *nnn* entity segments created.  
 Delete old polyline? (*Y or N*)
- **To fit cubic spline curve on CLAMPED condition**  
 @CMD: **CSPLINE** (*return*)  
 Select polyline: (*pick an open one*)  
 Cyclic/Relaxed/<Start point clamped direction>: (*point, angle or space bar*)

<End point clamped direction>: (*point, angle or space bar*)

Done! -- New polyline with *nnn* entity segments created.

Delete old polyline? (*Y or N*)

**Example:**

# CUTCLIP

## Cut Objects to Clipboard Command (Win)

---

### Purpose:

The **CUTCLIP** command is used to cut selected objects to the Windows Clipboard in various formats.

### Description:

The **CUTCLIP** command lets you copy selected objects to the Windows Clipboard and then erase them from the drawing. In fact, it is equivalent to execute a **COPYCLIP** command and followed by **ERASE** command. See **COPYCLIP** command for details.

### Procedure:

@CMD: **CUTCLIP** (*return*)  
Select Objects (+): (*do so*)  
Clipboard format -- Wrk/Dwg/Meta/Text/ALL <W>:  
...  
Insert Base Point: (*if Wrk/Dwg format specified*)

### Example:



# DBLIST

## List Drawing Database Command

---

### Purpose:

The **DBLIST** command is used to list out the content of the current drawing.

### Description:

The **DBLIST** command lets you list out the drawing from the beginning of the drawing database to the end. You may type **Ctrl/C** to stop the listing, and use the **Ctrl/Z** function to view the listing. This is an **AutoCAD**-compatible command.

Use **LIST** command to list out selected objects from the drawing, or use **ID** command to identify each element from the drawing. If your drawing are large, the listing generated by **DBLIST** will be very lengthy.

The purpose of this command is for debugging. For example, to identify whether there is something wrong in the drawing database that can not be seen from the screen (such as **INSERT** of undefined block, zero length objects, etc.), you may erase all the objects selected from the screen (don't use **ALL** option) and then **DBLIST**.

### Procedure:

Enter the **DBLIST** command to **TwinCAD** command prompt:

```
@CMD: DBLIST
```

and **TwinCAD** will list out all the drawing, until you type **Ctrl/C**.

### Example:

# DDFRAME

## Dialog Based Drawing Frame Creation Command (TCL)

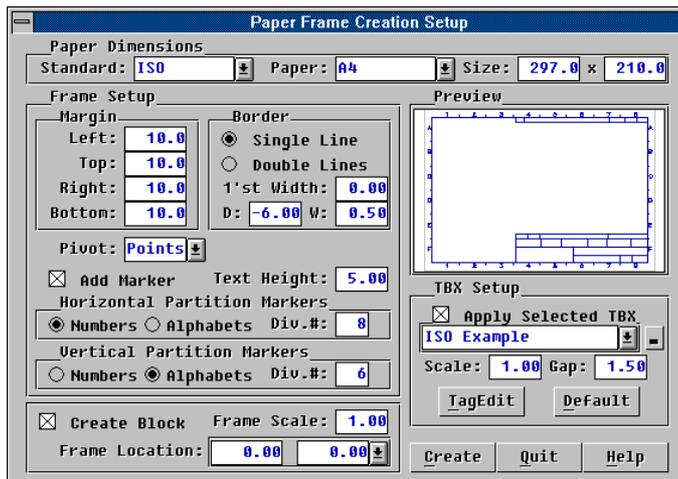
### Purpose:

The **DDFRAME** command is used to create a plain drawing frame based on a selected paper size and drawing scale.

### Description:

The **DDFRAME** command lets you create a plain drawing frame upon a selected paper size and drawing scale. You may specify the details of the frame style, frame location, and optionally choose to apply a selected **TBX** file to add title boxes on the drawing frame automatically. (A **TBX** file is an ASCII file that defines the creation of title boxes on a drawing frame. See latter sections.) You may choose to create the drawing frame as a block definition and make an insertion of it, or create it directly by drawing level entities.

Upon activation, the **DDFRAME** command will pop up a dialog window as shown in the figure below.



The following describes the details of these fields from the dialog window.

### Paper Dimensions

The use of a drawing frame to enclose the drawing view therein is for the paper output purpose. The size of the frame is therefore depends on the size of the paper intended to be used for the drawing output.

You may specify the size of the paper from the "Paper Dimensions" group, which contains the following items:

**Standard** A combo-box for you to select the paper standard. By picking at the box, you can select one paper standard to use from the drop-down list.

**Paper** A combo-box for you to select the paper size by its name under the current selected paper standard. By picking at the box, you can select the paper by its name to use from the drop-down list.

**Size** Two value entry fields for you to specify the size of the paper directly in units of mini-meter. These two entry fields also show the size of the paper being selected and are updated automatically when you change the specification of the paper.

## Frame Setup

The style of the drawing frame to create is setup from the group of "Frame Setup", from which you may specify the margin width of the frame with respect to the paper, the style of the frame border, the pivot marker of the paper, and the partition marker of the drawing frame, as described below.

### Margin

The distance from the paper edge to the outside border of the drawing frame is the margin of the frame. There are four of them from the "Margin" sub-group; namely, there are:

**Left** Value entry, specifies the left margin of the drawing frame in units of mini-meter.

**Top** Value entry, specifies the top margin of the drawing frame in units of mini-meter.

**Right** Value entry, specifies the right margin of the drawing frame in units of mini-meter.

**Bottom** Value entry, specifies the bottom margin of the drawing frame in units of mini-meter.

### Border

The basis of the drawing frame is the frame border. You may specify to draw the frame border with a single line in specific line width, or with two parallel lines in a given distance and each in different line widths, from the "Border" sub-group:

**Single Line** Radio button, selects the option to draw the border with a single line.

**Double Lines** Radio button, selects the option to draw the border with two parallel lines.

**1'st Width** Value entry, specifies the line width of the main border line in units of mini-meter.

**D** Value entry, specifies the distance of the second border line from the main border line in units of mini-meter, effective only when "Double Lines" option is selected. If this value is positive, the second line will be drawn inside of the main border. If it is negative, it will be drawn out-side of the main border.

**W** Value entry, specifies the line width of the second border line in units of mini-meter.

### Pivot of the Paper

The pivot of the paper marks the coverage area of the selected paper. With the pivot markers, you may easily map the drawing frame to the paper when the drawing is to output through

either **PRPLOT** or **PLOT** commands. Pick at the combo-box of "Pivot", and you may select one of the following pivot markers:

<b>None</b>	No pivot markers will be added to the frame.
<b>Points</b>	Two pivot markers of POINT entities will be added to the frame. One at the lower left corner and the other, top right corner of the paper area.
<b>Lines</b>	Each corner of the paper will be drawn with two lines perpendicular to each other around the corner as the pivot marker. The length of each line is determined by the corresponding frame margin.
<b>Box</b>	The paper area will be marked with a single rectangular box directly.

### Frame Partition Marker

The drawing frame may be virtually divided into several partitions in both horizontal and vertical directions. Along each direction, each partition will be assigned with a unique identifier which can be a number or an alphabet character. And thus, each small rectangle area determined by the partitioning in the drawing frame can be identified by addressing its partition identifiers in the both directions. This is useful for a person to locate a part drawing from the blue print by looking at a BOM table.

You may choose to create such a partitioning on the drawing frame by enabling the option "Add Marker". For each direction, you may specify either to use digital numbers or alphabet characters as the partition identifiers by selecting the corresponding radio button individually from each of the two sub-groups; namely, the "Horizontal Partitioning Marker" and "Vertical Partitioning Marker", respectively. The text height of the partition marker is specified by the value entry field "Text Height".

### TBX Setup

You may require to draw title boxes of specific layout on the drawing frame. The title boxes may also contain texts in some of the fields. More over, some texts may be variable and accessed as the attributes tagged to the drawing or to the drawing frame. Some fields in the title boxes may also contain symbols, company logos and various kind of informations.

No matter what kind of paper size you choose, each time a new drawing frame is to create, you may wish to apply the same or a different layout of title boxes on the drawing frame. All of this can be done by selecting an appropriate **TBX** file and enabling **DDFRAME** to apply it to create the title boxes for you.

A **TBX** file is an ASCII file that defines the creation of title boxes on a drawing frame. You may create or edit such a file via a simple text editor, such as the NotePad of Windows. Several **TBX** examples following some drafting standard are provided with **DDFRAME**, and you may easily modify one of them to suit your need. See later section for the details of **TBX** file description.

So, from the "TBX Setup" group, you may select an existing **TBX** file and choose to apply it to the drawing frame to create the desired title boxes for you automatically. The following describes the items in this group.

**Apply Selected TBX** A check box and a combo-box of **TBX** file selection. You may make a check in the check box to specify to apply the selected **TBX** file to create title boxes on the drawing frame, and pick at the combo-box to select the desired **TBX** file.

<b>Scale</b>	Value entry, specifies the scale factor of the title box defined by the <b>TBX</b> file.
<b>Gap</b>	Value entry, specifies the gap distance of text in a field from the field border in the title box. A positive value specifies an absolute distance which is also scaled by the scale factor. A negative value specifies the percentage of the field height.
<b>TagEdit</b>	Button, provides you the operation to edit the default contents of the attributes defined by the selected <b>TBX</b> file. If the current selected <b>TBX</b> file contains tag definitions and will apply it to the creation of the title boxes, this button will be enabled and you may press it to edit the default attribute contents.
<b>Default</b>	Button, lets you apply the default frame style setting from the current selected <b>TBX</b> file. The current selected <b>TBX</b> file may contain statements specifying the parameters for the frame style setting. You may press this button and apply these parameters to override current setting.

Note that there is a tiny button to the right of the **TBX** file combo-box. If you should create or add new **TBX** files to the **TwinCAD**'s directories, you may need to press this tiny button to collect them into the available **TBX** file list. **DDFRAME** will automatically search the **TwinCAD** directories to collect **TBX** file list for the very first time in its execution.

## Preview of the Drawing Frame

**DDFRAME** provides a preview window on the main dialogue window. Whenever you change the paper, frame style or select a different **TBX** file to apply, you will see the result instantly from the preview window. Note that, however, the preview does not show the details of texts in the title boxes created by applying the selected **TBX**.

## Drawing Frame Creation

An unnamed group located at the lower left corner of the dialog window provides the options of drawing frame creation.

<b>Create Block</b>	Check box, specifies whether to create the resulting drawing frame as a block definition and a block instance or as plain drawing entities. You may make a check in the check box to specify to create block. Note that the name of the block is always fixed in the form "sss-pp", where "sss" stands for the standard name and "ppp", the paper name. An example is "ISO-A4".
<b>Frame Scale</b>	Value entry, specifies the overall scale factor of the drawing frame.
<b>Frame Location</b>	Coordinate value entries, specifies the lower left corner of the paper location in the drawing file. The default is (0,0). If you press it by the drop-down button, you will be asked to designate the point from the drawing area.

## Other Buttons

You may press the **[Help]** button to see the helping text. Press the **[Create]** button to conclude the parameter setup and create the drawing frame. Press the **[Quit]** button to quit the command. Pressing <ESC> key also quits **DDFRAME**.

## Special Notes

**DDFRAME** collects **TBX** files only from **TwinCAD**'s installed directory, **COMMANDS** and **SUPPORT** sub-directories.

The **DDFRAME** command is an external command provided by the TCL program file "DDFRAME.TCL" or "DDFRAME.TCA". Either file must be present in the **COMMANDS** sub-directory where the **TwinCAD** resides. If you can not issue **DDFRAME** command, you may solve the problem by copying the "DDFRAME.TCL" to the **COMMANDS** sub-directory.

### **TBX files**

Please see Title Box Definition file (**TBX**) section.

### **Procedure:**

@CMD: **DDFRAME** (*return*)  
... [Dialog Operation] ...

### **Example:**

## Title Box Definition File (TBX file)

---

A **TBX** file is used to define the creation of title boxes on a drawing frame. It is a simple line-oriented ASCII text file containing a file descriptor and the following parts:

- Frame setup overrides statements
- TAG definitions
- FIELD definitions
- BOX definitions (containing ROW definitions)

<b>Paper-name</b>	Name of the paper. An asterisk (*) means the values shall apply to all size of the papers.
<b>left</b>	Left margin value, floating point, in units of mini-meter.
<b>top</b>	Top margin value, floating point, in units of mini-meter.
<b>right</b>	Right margin value, floating point, in units of mini-meter.
<b>bottom</b>	Bottom margin value, floating point, in units of mini-meter.

Examples are given below:

```
MARGIN: "*",10,10,10,10
MARGIN: "A0",15,15,15,15
MARGIN: "A1",15,15,15,15
MARGIN: "A2",15,15,15,15
MARGIN: "A5",5,5,5,5
MARGIN: "A5-P",5,5,5,5
```

## BORDER Statement

The BORDER statement setups the required frame border style of the drawing frame. The syntax is given below:

```
BORDER: "Paper-name",Type, W1 , D1, W2
```

where

<b>Paper-name</b>	Name of the paper. An asterisk (*) means the values shall apply to all size of the papers.
<b>Type</b>	Type of the border, string constant of either SINGLE or DOUBLE.
<b>W1</b>	Line width of the first border line, floating point in units of mini-meter.
<b>D1</b>	Distance between the first border line and the second border line, floating point in units of mini-meter.
<b>W2</b>	Line width of the second border line, floating point in units of mini-meter.

Examples are given below:

```
BORDER: "*", Double, 0, -6., 0.5
BORDER: "A0", Double, 0, -10., 0.5
BORDER: "A1", Double, 0, -8., 0.5
```

## MARKER Statement

The MARKER statement specifies the usage of partition marker of the drawing frame. The syntax is given below:

```
MARKER: "Paper-name", Th, Hdiv, Vdiv
```

where

<b>Paper-name</b>	Name of the paper. An asterisk (*) means the values shall apply to all size of the papers.
-------------------	--

<b>Th</b>	Text height, floating point in units of mini-meter. If this value is negative, the partition marker is turned off.
<b>Hdiv</b>	Division number of horizontal partitioning, integer. If it is negative, it means to mark with capital letter; otherwise, mark with digital number.
<b>Vdiv</b>	Division number of vertical partitioning, integer. If it is negative, it means to mark with capital letter; otherwise, mark with digital number.

Examples are given below:

MARKER: "\*" ,5.0, 8, -6  
 MARKER: "A0" ,7.0, 20, -14  
 MARKER: "A1" ,7.0, 14, -10  
 MARKER: "A2" ,7.0, 12, -8  
 MARKER: "A5" ,4.0,6,-4  
 MARKER: "A5-P" ,4.0,4,-6

## GAP Statement

The GAP statement specifies the default gap distance between text and field boundary. The syntax is given below:

GAP: "*Paper-name*", *gap*

where

<b>Paper-name</b>	Name of the paper. An asterisk (*) means the values shall apply to all size of the papers.
<b>gap</b>	Gap distance between text and field border, floating point. If it is positive, it specifies absolute gap distance in units of mini-meter. If it is negative, it specifies the relative gap distance in terms of the percentage of the field height.

An example is given below:

GAP: "\*" ,-5

## SCALE Statement

The SCALE statement specifies the required scale factor of the title box in the drawing frame. The syntax is given below:

SCALE: "*Paper-name*", *scale*

where

<b>Paper-name</b>	Name of the paper. An asterisk (*) means the values shall apply to all size of the papers.
<b>scale</b>	Scale factor of the title box dimension, floating point.

An example is given below:

SCALE: "\*" ,1.0

## TAG Definitions

The TAG definitions are optional in a TBX file. They are used to define the attributes of tags used in the TBX file and supply each of them a more expressive prompting string, which may help the operator to identify the meaning of the tag when he is asked to modify or to enter its contents.

The defining syntax is as below:

TAG: *Tag-name*,{D,}{V,} "*prompting string*"

It is the same as that appears in a TAG file produced by **TAGOUT** command. See **TAGIN/TAGOUT** command reference for the details about this TAG definition. In fact, you may use the preprocessing statement

#include "tag-file"

to include a TAG file. **DDFRAME** will automatically skip the GROUP definitions and load the TAG definitions therein.

Note that if a tag name is referenced without a corresponding TAG definition in the TBX file, **DDFRAME** will assume that it is a variable and visible attribute tag, and its name will be used as the prompting string.

## FIELD Definitions

The FIELD definitions are used to define the details of the text fields which are referenced in the ROW definitions that appear only in the BOX definitions. A field can be

- An empty field containing nothing, or
- A static text field containing a fix text string, or
- An attribute tag field containing an attribute tag to display variable text information, or
- A combination of a static text string used as the field display title and an attribute tag as the actual field content.

A FIELD definition must be completed within a text line. The general form of a FIELD definition is given below:

FIELD: *Field-name*, parameters ...

where *Field-name* is the ID name of the field.

### Static Text Field

The syntax for a static text field is as below:

FIELD: *Field-name*, "*text*" {, *Tadj*}

where

**text**                      Static text string as the field content. If text string within the pair of parenthesis is empty, then it is effective an empty field.

**Tadj**                      Optional Text justification code. It must be one of the following characters:

**L**                      Base-left justified.

<b>C</b>	Base-center justified.
<b>R</b>	Base-right justified.
<b>E</b>	Even-fitting within the field.
<b>F</b>	Direct fitting within the field.

An optional code string "90" can be added after the justification code to specify a vertical writing text.

Examples are given below:

FIELD: EMPTY,""

FIELD: COMPANY,"TCAM DEVELOPMENT HOUSE",F

### Attribute Tag Field

The syntax for an attribute tag field is as below:

FIELD: *Field-name*, *Tag-name* ,*Tadj*, "*Content*"

where

**Tag-name** Name of the attribute tag used in the field. If the specific TAG is not defined, it will be assumed to have both visible and variable attributes.

**Tadj** Text justification code.

**Content** Initial default value of the tag's content.

Examples are given below:

FIELD: OWNER,OWNER,C,"XXX"

FIELD: TITLE,TITLE,C,"XXX"

### Combination Field

The syntax for a combination field is as below:

FIELD: *Field-name*, *Tag-name* ,*Tadj*, "*Content*", *Wt*, "*Title*",*Txadj*

where

**Tag-name** Name of the attribute tag used in the field.

**Tadj** Text justification code of the tag.

**Content** Initial default value of the tag.

**Wt** Integer, title text field width specifier. If this value is positive, the field will be divided into two sub-fields horizontally, and the title text will be placed within the left one. The value specifies the width of the title sub-field in units of percentage of the total field width. A vertical line will be added for the separation of the two sub-fields.

If this value is negative, the field will be divided into two sub-fields vertically, and the title text will be placed within the top one. The value specifies the height of the title sub-field in units of percentage of the field height. Note

that, however, no additional line will be added to separate the two sub-fields explicitly.

<b>Title</b>	Static field title string.
<b>Txadj</b>	Text justification code of the title text.

Examples are given below:

```
FIELD: DESIGN,DESIGNEDBY,L,"XXX",-50,"Designed By",L
FIELD: CHECK,CHECKEDBY,L,"XXX",-50,"Checked By",L
FIELD: EDITION,EDITION,L,"0",-50,"Edition",L
FIELD: SHEET,SHEET,L,"1/1",-50,"Sheet",L
```

## BOX Definitions

The Box definitions defines the generation of the title boxes in the frame. You may have several BOX definitions within a TBX files.

The general syntax of a BOX definition is given below:

*BOX: Name, Ancho{/box-name}, Dir, Height, Width*

```
{
  Row Specification
  ...
  Row Specification
}
```

where

<b>Name</b>	Name of the box being defined. Since a box may be referenced by other boxes to obtain ancho position, a unique name must be given.								
<b>Ancho</b>	Ancho point ID number, from 0 to 3, as listed below: <table> <tr> <td><b>0</b></td> <td>Ancho to its lower left corner.</td> </tr> <tr> <td><b>1</b></td> <td>Ancho to its lower right corner.</td> </tr> <tr> <td><b>2</b></td> <td>Ancho to its upper right corner.</td> </tr> <tr> <td><b>3</b></td> <td>Ancho to its upper left corner.</td> </tr> </table>	<b>0</b>	Ancho to its lower left corner.	<b>1</b>	Ancho to its lower right corner.	<b>2</b>	Ancho to its upper right corner.	<b>3</b>	Ancho to its upper left corner.
<b>0</b>	Ancho to its lower left corner.								
<b>1</b>	Ancho to its lower right corner.								
<b>2</b>	Ancho to its upper right corner.								
<b>3</b>	Ancho to its upper left corner.								
<b>/box-name</b>	Optional name of the box to ancho. The slash code (/) is necessary to introduce the name of the box to ancho. If it missing, the overall drawing frame is assumed.								
<b>Dir</b>	Direction of the box extended from the ancho point, as described below: <table> <tr> <td><b>0</b></td> <td>Extended to the first quadrant.</td> </tr> <tr> <td><b>1</b></td> <td>Extended to the second quadrant.</td> </tr> <tr> <td><b>2</b></td> <td>Extended to the third quadrant.</td> </tr> <tr> <td><b>3</b></td> <td>Extended to the fourth quadrant.</td> </tr> </table>	<b>0</b>	Extended to the first quadrant.	<b>1</b>	Extended to the second quadrant.	<b>2</b>	Extended to the third quadrant.	<b>3</b>	Extended to the fourth quadrant.
<b>0</b>	Extended to the first quadrant.								
<b>1</b>	Extended to the second quadrant.								
<b>2</b>	Extended to the third quadrant.								
<b>3</b>	Extended to the fourth quadrant.								

<b>Height</b>	Integer, absolute height of the box, in units of mini-meter. If this value is zero, the height of the box will not be limited and will be the total height of each containing rows. However, if this value is not zero, the height of each containing row will be scaled such that the total box height meets the value.
<b>Width</b>	Integer, absolute width of the box, in units of mini-meter. In current release, this value can not be zero.
<b>{</b>	A single left brace in the next line after BOX statement introduces the content of the box.
<b>Row Specification</b>	Specification of a row within the BOX. Multiple row specifications must appear in a BOX definition in a from-top-to-bottom manner. See latter section for details of ROW specification.
<b>}</b>	A single right brace at the end of the last row specification concludes the definition of a BOX.

### Special Box Definition

If no ancho point is given after the box name, then the overall drawing frame will be the targeted box to define and will be treated specially. Rows in the box will be scaled to fit within the frame. This is useful in customization of drawing frame decoration.

An example is given below:

```
BOX: FRAME
{
    ROW: 1,@isoframe
}
```

The box will be the overall drawing frame; however, the outer profile of the box will not be drawn out, since it is treated specially. There is only one ROW specification in the box, so that the height of the row will be the height of the drawing frame. There is only one field specification in the row, so that the width of the field will be the width of the drawing frame. The vector generation program ISOFRAME.VEC (written in TCL) is referenced by the field to do the drawing frame decoration.

### ROW Specification

A ROW specification specifies the existence of a row and defines its content within a BOX definition. The syntax of a ROW specification is given below:

ROW: *Height, Field-Spec, ..., Field-Spec*

where

<b>Height</b>	Integer, height of the row in the box. The actual height of the row depends on the height specification of the BOX. If the BOX height is limited to an absolute value, the height of the row will be scaled accordingly.
<b>Field-Spec</b>	Specification of a field. It can be a direct static text, a field reference, an external VEC file reference, or a nested box inclusion. See latter paragraph for details.

## Field Specifications

The field specification specifies the existence of a field in a ROW from a BOX. It can be in one of the following form:

- Direct Text:  

$$"Text"/Width{Tadj}$$
- Field Reference:  

$$Field-name/Width{Tadj}$$
- External Vector Generation:  

$$@vec-name/Width{Tadj}$$
- Nested Box Inclusion:  

$$'{/Width{Tadj}}$$

Row Specification

...

Row Specification

$$'',{ next Field-spec continued}$$

where

<b>Text</b>	Direct text string, enclosed in parenthesis, to be displayed in the field. It is convenient to use direct text than a reference to the FIELD definition of static text. However, except the ROW containing fields of nested box inclusion, a ROW specification can not span over a text line, which can have at most 255 characters in length. In some title box designs with long rows, you may still need to use the FIELD definitions of static text.
<b>Field-name</b>	Name of an existing FIELD definition. The FIELD must be present before it is referenced.
<b>vec-name</b>	Name of the vector generation file (a special TCL program in VEC file extension) to be executed to give the content of the field.
<b>Width</b>	Integer, width of the field within the row. If this value is negative, it specifies the percentage of the row width. If it is zero or missing, it specifies to take up the rest of the row space, effective only when it is the last field in the row.
<b>Tadj</b>	Text justification code override. See FIELD Definitions for text justification code format.
<b>{</b>	Start of a nested box inclusion. The next line should be the content of the nested box definition.
<b>}</b>	End of a nested box inclusion and continue the current ROW definition.

Examples are given below:

```
FIELD: NOTE,"Title/Name, Designation, material, dimension etc",L
```

```

BOX: Main,1,1,0,170
{
  ROW: 6,"Itemref"/15C,"Quantity"/17C,NOTE/88L,"Article No/Reference"/50L
  ROW: 12,DESIGN/32,CHECK/25,APPROV/33,FILENAME/25,DATE/30,SCALE/20
  ROW: 20,OWNER/75,{/
    ROW: 10,TITLE
    ROW: 10,DWGNO/60,EDITION/15,SHEET
  }
}

```

## Conditional Statements

**DDFRAME** supports the following conditional statements in the TBX file:

- **#ifpaper** -- Test if a specific paper is selected. An argument of quoted string as the name of paper to test must be given. If the test is successful, the contents that follows the statement will be effectively loaded in; otherwise, it will be ignored until the next conditional statement is encountered.
- **#ifnot** -- Test if a specific paper is not selected. An argument of quoted string as the name of paper to test must be given. If the test is successful, the contents that follows the statement will be effectively loaded in; otherwise, it will be ignored until the next conditional statement is encountered.
- **#else** -- If the previous test is not successful, then the contents that follows this statements are loaded in until the next conditional statement is encountered.
- **#endif** -- End of the if-then-else construct.

## Include statements

You may include an external file, such as TAG file, from the TBX file by the syntax:

```
#include "filename"
```

where *filename* is the name of the file to include.

## The VEC file (vector generation file)

A vector generation file is a TCL program file that is responsible to generate a vector drawing fitted within a given rectangle defined by the caller, say **DDFRAME**.

It is called with the arguments passed through the TCL registers:

- P1** Lower left corner of the rectangle.
- P2** Upper right corner of the rectangle.
- I0** Justification code if applicable.
- V0** Gap distance of Text.
- V1** Total scale factor.
- S0** ID code, always be "VEC".
- S1** Current selected paper name.

# DDHATCH

## Associate Hatch Creation and Modification Command (TCL)

### Purpose:

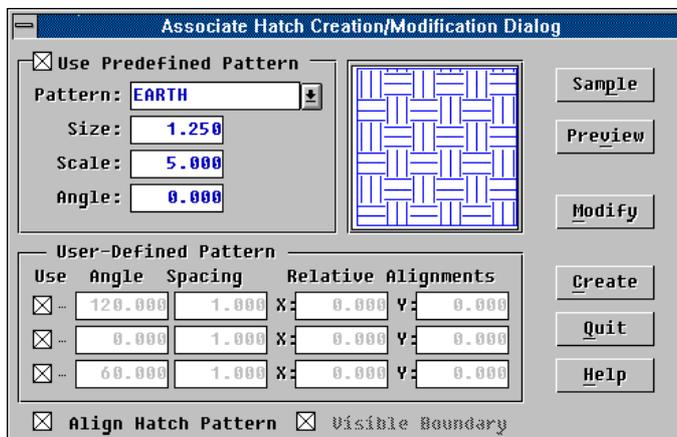
The **DDHATCH** command is used to create an associate hatch region or to modify the associate hatch state of selected region entities, via a GUI dialogue operation manner.

### Description:

The **DDHATCH** command lets you setup the required crosshatch pattern and then create an associate hatch region or replace the associate hatch of selected regions with it. The same functionalities are also provided by the **HATCH**, **RHATCH** and **CHANGE** commands. However, **DDHATCH** provides a much better user interface in specifying the crosshatch pattern.

**DDHATCH** will create and maintain its own private hatch pattern table out from the current pattern file used by **TwinCAD**, and a slide library containing slides corresponding to these patterns. If these informations can not be found from the installed directory or are already out-of-date, **DDHATCH** will begin an initialization process first before entering the service dialogue at its start-up. See latter section for more information about the initialization process.

If everything is fine, **DDHATCH** will pop up the dialogue window as show below:



You may setup the required crosshatch pattern by selecting one from the predefined patterns or by giving direct hatch line specifications, and then click at the **[Create]** button to create a new hatch region to associate with it, or click at the **[Modify]** button to select existing regions and to replace their associate hatch with it.

The following describes the details of these screen items from the top of the dialogue windows.

### Use Predefined Pattern

This group title is also an option check box. You may pick at it to toggle its checking state. If this option is enabled (i.e. being checked), the predefined pattern specified in this group will be used for the crosshatch and those items in this group will become effective:

- Pattern** A combo-box type value field, specifying the name of the predefined pattern to use. You may pick it by its drop-down button to select a name directly from a list of predefined patterns. Or, you may pick it by its value box to choose the desired pattern from the pop-up icon menu, as shown in the Example section.
- Size** A value entry field, specifying the size of the pattern in terms of the Minimum Y-interval of the repeated lines in the pattern. Change this size value will also update the scale factor.
- Scale** A value entry field, specifying the scale factor of the pattern. Change this scale value will also update the size value.
- Angle** A value entry field, specifying the rotation angle of the pattern in units of degrees.

The selected pattern will be shown in the preview sub-window with a slide from the "DDHATCH.SLB" slide library, provided that it is present. Note that the effect of the pattern scale and rotation angle is not reflected from the preview image. To observe the true-size image of the resulting pattern, you may click the **[Preview]** button. See latter paragraphs on the **[Preview]** button.

## User-Defined Pattern Group

If the use of the predefined pattern is disabled, then user-defined pattern will be used and the items from this group will become effective. A user-defined pattern is a pattern defined by direct hatch line specifications. Each hatch line specification contains the following three parts:

- |                            |   |
|----------------------------|---|
| <b>Angle</b>               | Direction angle of the hatch line in units of degree.   |
| <b>Spacing</b>             | Offset distance between each repeated hatch lines in drawing unit. This value can not be zero.  |
| <b>Relative Alignments</b> | Relative position with respect to a given alignment point that one of the repeated hatch lines must pass through. This consists of two separate value entry fields, one for the delta-X and the other, delta-Y. |

For an associate hatch region using user-defined pattern, there can be at most three distinct hatch line specifications. These are tabulated in this group. You can modify any of these control values by clicking at the corresponding entry field. However, to make a hatch line specification become effective, you have to check at its corresponding check-box.

The resulting pattern will be shown in the preview sub-window, provided that the use of predefined pattern is disabled. Note that the true size of the pattern with respect to the current drawing is determined by the spacing between hatching lines, and is not reflected by the preview image. To observe the true-size image of the resulting pattern, you may click the **[Preview]** button. See latter paragraphs on the **[Preview]** button.

## Align Hatch Pattern

This is an option check box. If this option is enabled, you will be asked to designate an alignment point when you click at the **[Create]** button to create a new associate hatch region. **DDHATCH** will prompt:

Please designate pattern alignment base point:

The pattern alignment point is a point through which the pattern origin will pass. The default will be at point (0,0).

## Visible Boundary

This is an option check box. If this option is enabled, the associate hatch region created will have a visible boundary; otherwise, the boundary will be invisible.

All associate hatch regions created by **HATCH** command are with invisible boundaries.

### The [Sample] Button

You may click this button to sample an existing region entity from the drawing to setup the current hatch specification. You will be asked to pick up the region entity by the prompt:

Please select the hatch region to sample:

**DDHATCH** will read the associate state of the region and set the current hatch specification accordingly.

### The [Preview] Button

You may click this button to view the physical size of the crosshatch pattern that appears in the preview sub-window from the drawing. **DDHATCH** will create a box of crosshatch and prompt

A 6-unit crosshatch is created at lower left corner...

Press space bar to end this crosshatch size preview:

You may enter **ZOOM** and **PAN** commands transparently to compare the crosshatch size with respect to your current drawing or the targeted boundary area.

Although the appearance of the current hatch pattern specification, either from predefined pattern or from direct hatch lines, can be observed from the preview sub-window, the relative size of the pattern to the current drawing

## DDHATCH Initialization

**DDHATCH** requires additional informations from the following two disk files:

- Pattern Table: A file containing a compacted informations about the current loaded hatch patterns in **TwinCAD**. It must have the same path name as the current loaded pattern file but with a different file extension as "DDH".
- Slide Library: A file containing corresponding slides to these predefined patterns. It must be "DDHATCH.SLB" and be located in one of the search path of **TwinCAD**.

If **DDHATCH** can not find and load in the Pattern Table, or the Pattern Table is out-of-date because the pattern file had been updated, **DDHATCH** will issue the message:

```
** Initialize DDHATCH ...
```

and begin to create a new pattern table.

After the pattern table is created or loaded in, **DDHATCH** will search for the "DDHATCH.SLB" slide library and check to see if there are corresponding slides in it to each of the predefined hatch patterns. If this slide library is missing, or there is at least one slide missing (maybe due to new pattern definition added to the pattern file), **DDHATCH** will pop up a query dialogue asking you if you want **DDHATCH** to build one or to add new slides for you.

Note that, without the corresponding slide, you won't be able to view the predefined pattern from the preview sub-window. However, if you like, you may create the required slides using **MSLIDE** command and update the "DDHATCH.SLB" slide library using the DOS utility **ASLIDE.EXE** all by yourself.

## Creating Slides

If you permit, **DDHATCH** will create slides for each corresponding predefined hatch patterns and save them into the "DDHATCH.SLB" slide library automatically. This requires the DOS utility **ASLIDE.EXE** be located from the search paths.

Since the aspect ratio of the slide to create has better be the same as that of the preview sub-window which is about square, you may have to adjust the window frame of **TwinCAD** such that the drawing window is about square for better slide generation.

**DDHATCH** will prompt

```
** Before generating the required slide ...
```

```
Please adjust the drawing window to about square:
```

You may press the space bar to proceed the slide generation.

## Special Notes:

The **DDHATCH** command is an external command provided by the TCL program file "DDHATCH.TCL" or "DDHATCH.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **DDHATCH** command, you may resolve the problem by copying the "DDHATCH.TCL" to the COMMANDS sub-directory.

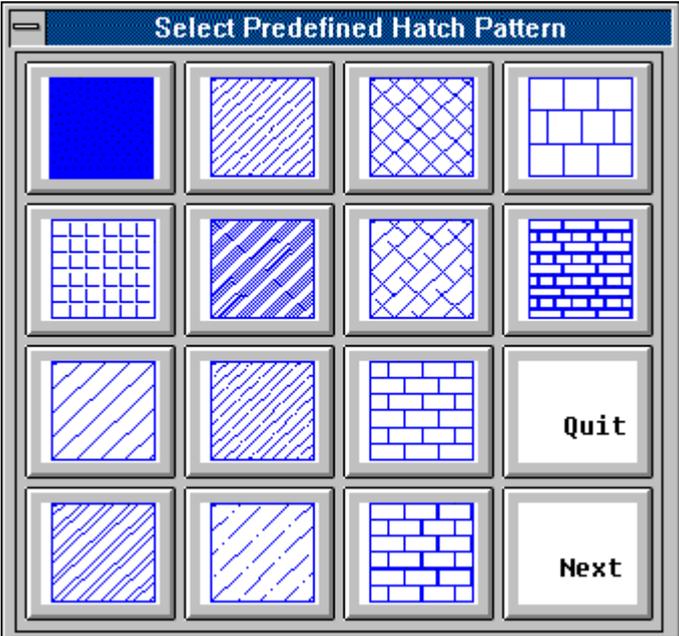
## Procedure:

```
@CMD: DDHATCH (return)
```

```
...[Dialogue Operation]
```

## Example:

The following is an example icon menu for pattern selection:



**DDIM**

## Diametric Dimensioning Command

---

**Purpose:**

The **DDIM** command is used to create a diametric dimension.

**Description:**

The **DDIM** command lets you dimension an arc or a circle by its diameter. You pick up the arc or the circle to dimension, and **TwinCAD** generates the dimensioning for you. Dragging of the dimensioning result is provided and is started for you to decide where to place it, as soon as you pick up the arc or circle.

The dimension text is generated automatically in the drawing unit to reflect the true diameter of the object. The generation of this default dimension text is governed by the dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about dimensioning.

The dimension style is determined by the position of the dimension text during the interactive dragging. The dimension line will be inside of the circle on the diameter, if the dimension text is dragged inside, and it will be outside, if the text is outside. The change can be seen directly from the dragging operation.

During the dragging operation, you may enter a direct angle value in units of degrees to specify the direction angle of the dimension line required for the dimensioning. The dimension text will then be placed at about the same distance from the current dragging position to the center of the dimensioning.

If the accuracy of the angle direction of the dimension line is of no concern, you may simply designate a point to settle down the placement of the dimensioning. The dragged image serves as a preview of the dimensioning when you designate the point by the current cursor position.

Before settling down the placement of the dimensioning, you may modify the dimensioning by the following sub-command options:

- C** Change text, specifying to change the dimension text manually. **TwinCAD** will prompt:

Dimension Text:

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

- D** Default text, specifying to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about it.
- OD** Dimension-Line Option, specifying to toggle the current effect of dimension line option. The dimension line option is effective only when the dimension text is dragged outside of the dimensioned circle. It specifies whether to generate the

dimension line (with arrow) inside of the circle while the dimension text is outside of it.

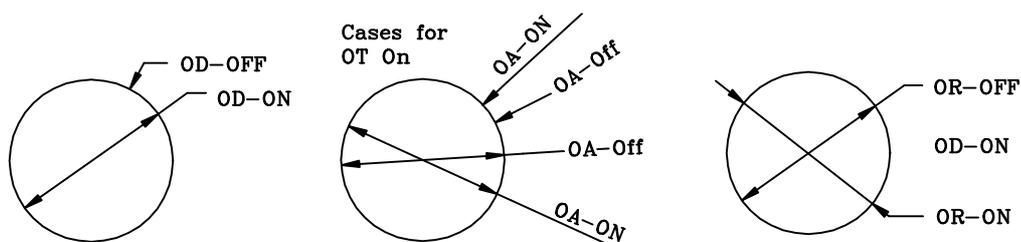
If this option is OFF, the dimension line and arrow will be generated outside pointing to the circle. If it is ON, the dimension line will be extended across the diameter of the circle and the arrow pointers are generated as if the dimension text is placed inside of the circle. This option can be preset at bit 0 of the dimension variable **DIMDOPT**.

- OT** Text generation option, specifying to toggle the current effect of text generation option. It specifies whether to align the dimension text with the dimension line or not. If this option is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the direction as the dimension line. This option can be preset at bit 1 of the dimension variable **DIMDOPT**.

If the dimension text is placed outside of the circle and this option is OFF, an additional lead line will be added to lead the horizontal text to the dimension line. The length of this lead line is determined by the dimension variable **DIMDRLEAD**.

- OA** Text alignment option, specifying to toggle the current effect of text alignment option. It specifies whether to align the dimension text above the dimension line or not. If this option is OFF, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. If it is ON, the text will be placed above the dimension line (and the dimension line will be extended to cover the span of text). This option can be preset at bit 2 of the dimension variable **DIMDOPT**.

- OR** Dimension line reverse option, specifying to toggle the current effect of dimension line reverse option, which is effective only when the dimension text is dragged outside of the dimensioned circle and the OD option is ON. It specifies whether to reverse the dimension arrow pointer direction when the OD option is ON. If this option is ON and is effective, then the arrow pointers that were generated inside of the circle will be generated outside pointing into the circle. The dimension line will also be extended over the circle for the additional arrow pointer outside of it. This option can be preset at bit 3 of the dimension variable **DIMDOPT**.



## Auto-extension-Arc Feature

When you are dimensioning an arc with the dimension generated inside of the arc, and if the span of the arc is less than  $180^\circ$ , then at least one arrow point of the dimension line will point to a position outside of the arc span. When this happens, the auto-extension-arc feature will be activated and an extension arc (see example) will be generated for dimensioning purpose. In fact, this auto-extension-arc feature will always be activated whenever the dimension arrow pointer points outside of the dimensioned arc.

## Dimensioning on PUCS-Plane

The **DDIM** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection, and allows you to dimension a projected circle (Ellipse) on the projected plane from the virtual space directly.

If the current **PUCS**-plane is active and valid for an Axonometric Projection, you may pick up ellipses to create projected diametric dimensionings directly, according to the rules below:

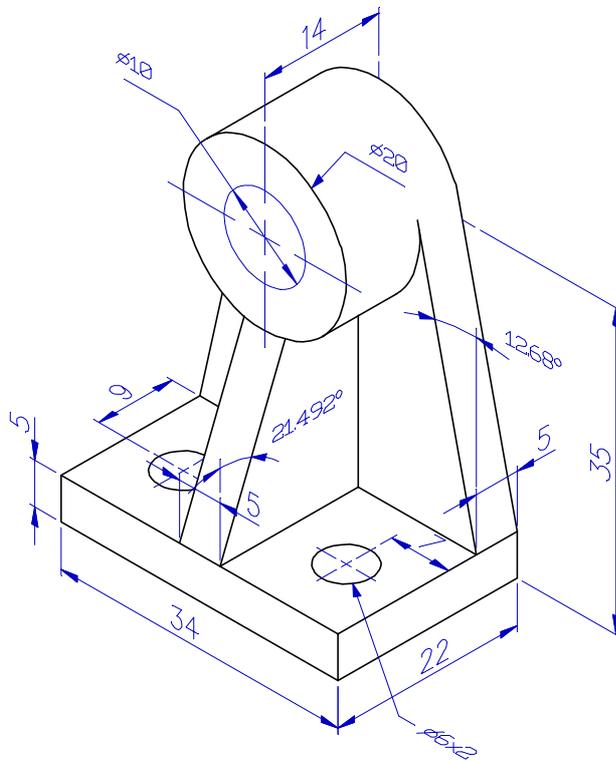
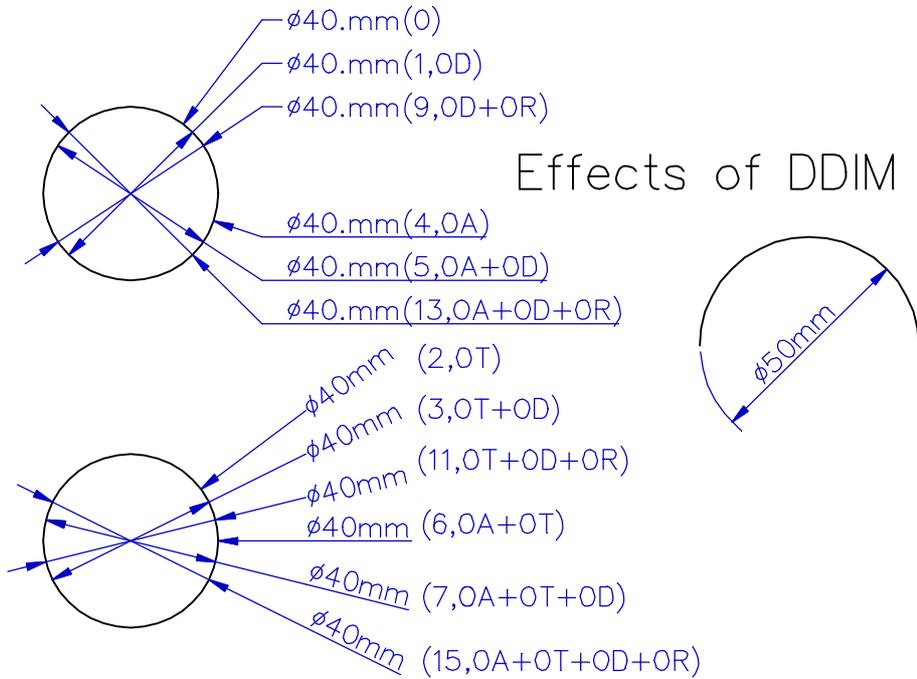
- The ellipse to dimension must be a one that can be produced by using current Axonometric Projection setup; otherwise, **TwinCAD** will not accept it for the operation, since it can not be used to determine a circle in the virtual space from the projection relationship.
- The dimension text is always aligned with the projected X-axis direction.
- The dimension value (diameter/radius of the circle) will be determined based on the current axonometric axes setup and the size of the ellipse.
- Once it is accepted and the dragging is started, the **PUCS**-plane will be locked temporary until the dimensioning is done. During the settling of the dimension, however, you may switch the axes setup of the UCS-plane around its 4 possible orientations, using the **<Ctrl/I>** function. The axes setup orientation of the UCS-plane will affect the projected image of the dimensioning. The dragging image will dynamically change upon the changing of the UCS-plane's axes setup.

You may select a closed polyline that was created by **ELLIPSE** command to approximate an ellipse. The **DDIM** command is able to check if it is an approximation of an ellipse, and extracts out the ellipse information

**/ITRYS**

Default-text/Change-text/OT(Text option)/OR/OD(Dimension-line option)/  
 OA(Adjust-Option)/<Set dimension text position or direction:(*point or value*)

**Example:**



Dimensioning Under PUCS-Plane

# DDINSERT

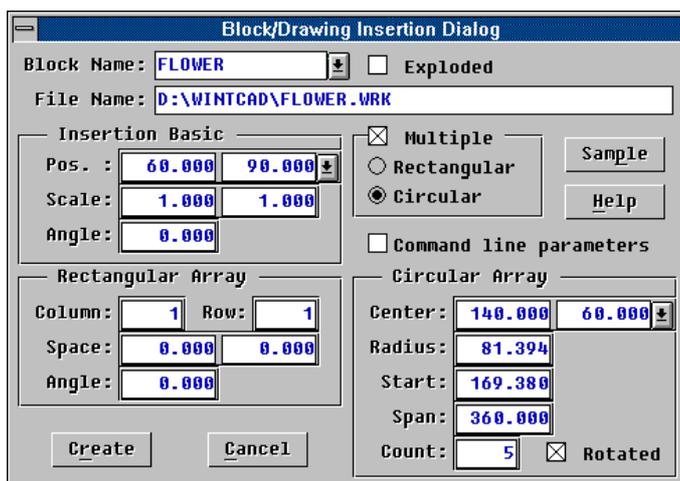
## Dialogue Based Block Instance Insertion Command (TCL)

### Purpose:

The **DDINSERT** command is used to create a block instance (a reference to a block) or multiple block instances in a rectangular or circular array pattern via a GUI dialogue operation manner.

### Description:

The **DDINSERT** command lets you create a block instance or a group of block instances in a rectangular or circular array pattern, providing the same functions as those provided by **INSERT**, **MINSERT** and **RINSERT** commands. However, unlike these **INSERT**, **MINSERT** and **RINSERT** commands, which require you to give the parameters step by step from the command line, **DDINSERT** lets you setup these parameters from a GUI dialogue window, as shown below, and create the block instance at one single click of button.



A block instance is a reference to a predefined drawing block. It can be scaled and rotated as required, and inserted at any designated position in the drawing. It can be made as a reference either to an existing Block in the current drawing or to an external drawing in WRK/DWG format. Yet, in the latter case, the external drawing is loaded and grouped as a Block with a specified reference name, before the block instance is made.

A group of block instances in rectangular or circular array pattern is not really an array of block instances, but a single entity containing multiple block instance references and called a multiple block instance. The informations related to the rectangular or circular patterns are part of that entity.

The operation of **DDINSERT** is simple. Setup the required parameters and then click the [**Create**] button to complete the job. The following describes the details of these screen items from the dialogue window.

### Block Name

This is a combo-box entry field specifying the name of the block to reference. If you pick it by its drop-down button, you can select the block name from the list of existing blocks in the drawing. If you pick it by its entry field, you will be asked to enter the block name directly into

the field. You will also be asked to enter the name directly, if you have selected the <Drawing> entry from the drop-down list.

If the block name entered in this field does not exist in the drawing, it must be an external drawing insertion. **DDINSERT** will automatically search for the drawing file of the same name from the search paths as the file to insert. If it can not be found, you will be asked to specify it explicitly via the pop-up file window.

## File Name

This is a data field containing the full path name of the drawing file to insert. Pick at this field and you will be asked to select the drawing file to insert via the pop-up file window. To remove the setting of this drawing file, you will have to cancel the pop-up file window.

The default block name used for the drawing file to insert will be the name of the file. However, if you have already specified an existing block name to reference, this would cause that existing block be redefined by the external drawing when the block instance is created. **DDINSERT** will ask you to confirm this when this situation happens. If your confirmation is negative, **DDINSERT** will automatically reset the block name to the default.

## Exploded

This is an option check box. Check this option if you want the block instance be exploded automatically as if the insertion were a copy of the original block definition instead of a reference to it. Note that if the insertion is from an external drawing, then no new block definition will be created if this option is enabled.

## Multiple

This is an option check box as well as a group title. Check this option if you are going to create a multiple block instance. There are two type of multiple block instance; namely, they are

**Rectangular** The group of block instances are in rectangular array pattern.

**Circular** The group of block instances are in circular array pattern.

You can click the corresponding radio button from this group to select the type of multiple block instance to create.

## Command line parameters

This is an option check box. Check this option if you are going to supply all the required parameters from the command line at the creation of the block instance. See **INSERT**, **MINSERT**, **RINSERT** for the respective command line operation procedure.

If this option is enabled, all the parameters from "Insertion Basic", "Rectangular Array" and "Circular Array" group will be disabled. **DDINSERT** will let you follow the command prompt to supply these required parameters when you click the [**Create**] button.

## Insertion Basic Group

The Insertion Basic group contains basic parameters for the block instance creation:

**Pos.** Two value entry fields with a drop-down button, specifying the X and Y coordinates of the basic insertion point, respectively. The basic insertion point is where the block instance will be inserted. You may click the drop-down button to designate the point

from the drawing screen. This point is also the base insertion point of the block instance from a rectangular or circular array.

- Scale** Two value entry fields specifying the scale factor of the block instance in the X and Y direction, respectively.
- Angle** A value entry field specifying the rotation angle of the block instance with respect to the X-axis direction in units of degrees.

### Rectangular Array Group

The Rectangular Array group contains additional parameters to those from the Insertion Basic group for the multiple block instance to be created in rectangular array pattern:

- Column** An integer entry field specifying the number of columns of the rectangular array.
- Row** An integer entry field specifying the number of rows of the rectangular array.
- Space** Two value entry fields specifying the space between columns and rows in drawing unit, respectively.
- Angle** A value entry field specifying the base angle of the array with respect to the X-axis direction in units of degrees.

### Circular Array Group

The Circular Array group contains additional parameters to those from the Insertion Basic group for the multiple block instance to be created in circular array pattern:

- Center** Two value entry fields with a drop-down button, specifying the X and Y coordinates of the circular array center, respectively. You may click the drop-down button to designate the center point from the drawing screen.
- Radius** A value entry field specifying the radius of the circular array, which is the distance between the array center to the insertion point of the first block instance from the array.
- Start** A value entry field specifying the starting angle of the first block instance from the array with respect to the array center in units of degrees.
- Span** A value entry field specifying the spanning angle of the circular array with respect to the array center in units of degrees.
- Count** An integer entry field specifying the number of block instance elements will be in the circular array.
- Rotated** An option check box specifying whether to rotate each of the block instance element such that the relative orientation of the block instance with respect to the array center is constant. If this option is not enabled, each block instance element of the circular array will have the same absolute orientation with respect to the X-axis. However, if it is enabled, each element will have the same relative orientation with respect to the array center, though the absolute orientations are different.

### The [Sample] Button

You may click this button to sample an existing block instance from the drawing to setup the current parameters for the new block instance to create. You will be asked to pick up the block instance by the prompt:

Please select the block instance text to sample:

Note that all the parameters will be updated by the block instance you have picked from the drawing except the basic insertion point.

## The [Create] Button

After you have setup proper parameter values, you may click this button to create the block instance. If you have checked the "Command line parameters" option, you will be asked to supply the required parameters by the command prompts from the command area. See also **INSERT**, **MINSERT** and **RINSERT** for details.

## Other Buttons

You may press the **[Help]** button to see the helping text. Press the **[Cancel]** button to end **DDINSERT** immediately. Pressing <ESC> key also quits **DDINSERT**.

## Special Notes:

The **DDINSERT** command is an external command provided by the TCL program file "DDINSERT.TCL" or "DDINSERT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **DDINSERT** command, you may resolve the problem by copying the "DDINSERT.TCL" to the COMMANDS sub-directory.

## Procedure:

@CMD: **DDINSERT** (*return*)  
...[Dialogue Operation]

## Example:

# DDNEW

## Create A New Work Drawing File Command (TCL)

---

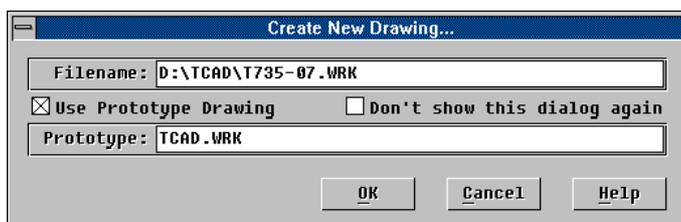
### Purpose:

The **DDNEW** command is used to create a new work drawing file that is completely empty or pre-loaded with a specific prototype drawing.

### Description:

The **DDNEW** command lets you create a new work drawing file via a GUI dialog window operation. You may specify to create a completely empty work drawing file or to pre-load it with a specific prototype drawing. Moreover, you may specify not to show up the main dialog window but to go on the creation of the new drawing file directly. This helps simplify the command procedure, once it is well setup.

If the dialog window is enabled, **DDNEW** will pop up the dialogue window as shown below:



The following describes these screen items in details.

<b>Filename</b>	Filename string entry field, specifying the path name of the new drawing file to create. Click at this entry field and you will be asked to enter the filename via the general file window operation.
<b>Prototype</b>	Filename string entry field, specifying the path name of the prototype drawing file to use. Click at this entry field and you will be asked to select a work drawing file used as the prototype drawing via the general file window operation.
<b>Use Prototype Drawing</b>	Check box, indicating whether the <b>DDNEW</b> will pre-load a prototype drawing file for the new created drawing file or not. If this check box is marked, it will. You may click it to change its check state.
<b>Don't show this dialog again</b>	Check box, indicating whether the <b>DDNEW</b> will show the same dialog again at the next <b>DDNEW</b> command invocation or not. You may click it to change its check state. If this check box is marked, <b>DDNEW</b> will go on the creation of the new drawing file directly without entering this dialog window again at its next invocation. It will directly ask for the new drawing filename to create and will create it using current prototype drawing setup.

### Re-enable the Show-up of Dialog Window

Once you have disabled the show-up of the dialog window, **DDNEW** will directly pop-up file window for you to specify the new drawing file to create whenever the **DDNEW** command is entered. However, if you quit the file window without specifying any drawing file to create, the dialog window will show up again, where you may change the prototype drawing setup.

### Check to Save Current Drawing File

Before actually creating the new drawing file, **DDNEW** will check to see if the current drawing file has been modified or not. If it has been modified, **DDNEW** will pop up a query box asking you whether to save it or not, and save it if the answer is positive.

Note that a drawing is said to have been modified if there has been at least one command executed after the drawing was loaded or saved.

### Special Notes:

The **DDNEW** command is an external command provided by the TCL program file "DDNEW.TCL" or "DDNEW.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **DDNEW** command, you may solve the problem by copying the "DDNEW.TCL" to the COMMANDS sub-directory.

### Procedure:

```
@CMD: DDNEW (return)  
[Dialog windows operation...]
```

...

### Example:

# DDSTYLE

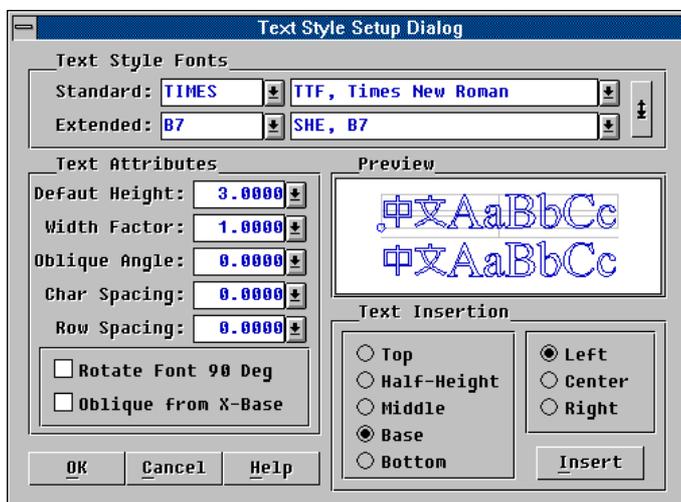
## Text Style Dialogue Setup Command (TCL)

### Purpose:

The **DDSTYLE** command is used to setup or modify the current text style parameters for subsequent **TEXT** creations in a GUI dialogue manner.

### Description:

The **DDSTYLE** command lets you setup or modify the current text style parameters for subsequent **TEXT** creations in a more convenient GUI dialogue manner than that provided by the **STYLE** command. **DDSTYLE** will pop up a dialogue window as shown below:



### Text Style Fonts

There are two parameter items in the "Text Style Fonts" group; namely, they are

- Standard** Corresponding to the text style variable **TXTSTYLE**, this parameter specifies the main text style name used for the text in ASCII codes.
- Extended** Corresponding to the text style variable **TXTESTYLE**, this parameter specifies the extension text style name used for the text in non-ascii codes, such as the two-byte coded Chinese, Japanese and Korean, and multi-byte coded Thai.

There are two combo-boxes for each of the above parameter setups. The shorter one on the left shows the name of the text style, which is the name used in the drawing. And, the longer one on the right shows the type and the name of the font, if the font name is available. Both of them provide the same selection set of the available fonts in the system, though in different view of the fonts. Changing either one of them will update the other.

### The Loadable Font Table

When you pick at any combo-box in "Text Style Fonts" group, you will be able to select a font from the dropdown list-box of available fonts. These font informations are read from a

Loadable Font Table named "DDSTYLE.SET", which was built by **DDSTYLE** and can be rebuilt anytime.

If you have installed new fonts into the system and want **DDSTYLE** to be aware of them and to include them in the Loadable Font Table, you may press the tiny unnamed button on the right-most side of the "Text Style Fonts" group. This will force **DDSTYLE** to rebuild the Loadable Font Table.

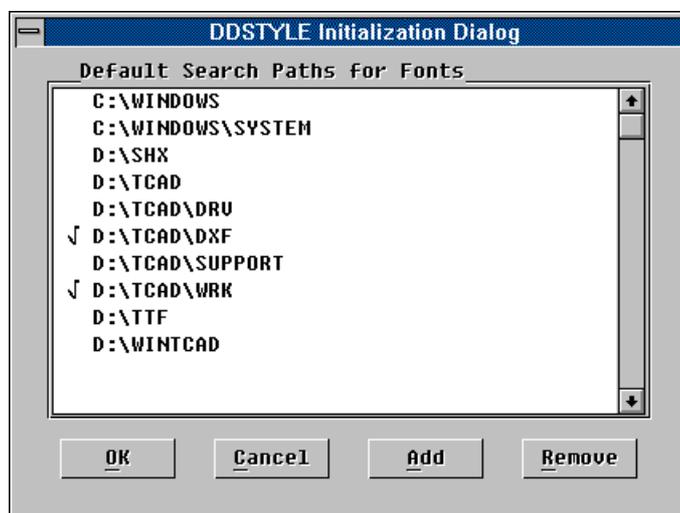
In fact, if **DDSTYLE** can not find the table upon execution, e.g. at the very first time execution since installation, it will automatically proceed the operation to build the Loadable Font Table.

## Building Loadable Font Table

**DDSTYLE** will automatically search known directory paths for available fonts and build the required Loadable Font Table. It will take the following paths as the default search paths:

- Windows directory and Windows System directory,
- Paths specified by the **TCADPATH** environment variable,
- Paths specified by the **ACAD** environment variable,
- Paths specified by the **TCADPATH** parameter entry from the **TwinCAD's** system profile file (INI file).

However, before doing the search, **DDSTYLE** will pop up a search path setup dialogue window for you to modify the search paths from the above default. You may remove some undesired search paths, and add some new search paths (such as where the new fonts are installed). An example of the search path setup dialogue window is given below:



To remove undesired paths, you must select them first and then press the [**Remove**] button. To select or un-select a path, simply pick at it. A check mark will appear before a selected path.

To add a new path, press the [**Add**] button. You will need to browse the file window and select the directory path to add in.

To start building Loadable Font Table, press the [**OK**] button. **DDSTYLE** will start searching the following supported font files:

- Windows(R) TrueType Font Files (TTF)

- Adobe PostScript(R) Type 1 Font (PFB)
- **AutoCAD**(R) Shape Font File (SHX)
- **TwinCAD** Shape Font File (SHZ)
- **TwinCAD** Extended Shape Font File (SHE)

Note that duplicated font files will be removed from the available list. This include those font files driven by SHE driver font files.

**DDSTYLE** will report the progress in the command area during the search of font files, and return to the main dialogue window when it finishes the building job.

## Text Attributes

**TwinCAD** supports many text control attributes. The following are those supported by **DDSTYLE**:

- Default Height** Corresponding to the text style variable **TXTHEIGHT**, this parameter specifies the size of the text font in height. If this value is negative, the text will be mirrored with respect to the base writing line.
- Width Factor** Corresponding to the text style variable **TXTWIDTHF**, this parameter specifies the width factor of the text font to generate. It is usually 1. If this value is negative, the text will be mirrored to the opposite of the writing direction.
- Oblique Angle** Corresponding to the text style variable **TXTOBANGLE**, this parameter specifies the oblique angle of the text font to generate, in units of degrees. It is usually 0. Positive value slant the font backward, and negative value slant it forward. However, if the attribute to "Oblique from X-Base" is enabled, positive value will slant the font up-ward, and negative value, downward.
- Char Spacing** Corresponding to the text style variable **TXTCSPACE**, this parameter specifies the character space adjustment in text font generation. A positive value specifies absolute space (additional to the original one defined by the font), negative value specifies relative space ratio over the text height. A zero value will disable this adjustment function.
- Row Spacing** Corresponding to the text style variable **TXTRWDIST**, this parameter specifies the default text row distance. See **STYLE** command for more details.

The above parameter setups are provided in combo-box. You may pick it by the dropdown button to select a value directly from those of common use. However, if you pick it by the value box, you will be asked to enter the value directly.

Besides the above value control attributes, there are also on-off control attributes provided in check boxes:

- Rotate Font 90 Deg** Corresponding to the text style variable **TXTROTCCW**, this on-off state selection specifies to rotate the text font by 90° Counter-ClockWise or not. This attribute is useful in writing Chinese/Japanese/Korean texts or the likes in vertical manner.

**Oblique from X-Base** Corresponding to the text style variable **TXTOBX**, this on-off state selection specifies to oblique the font from the its base line or not, if the oblique angle is not zero. If it is off, the text font will be obliqued in the usual way. However, if it is on, the font will be obliqued from its base line, suitable for text font in vertical writing.

### Preview of Text Style Font

**DDSTYLE** provides a preview window on the main dialogue window. When you change the text style font or the text control attributes, you will see the result instantly from the preview window displaying the example text strings.

From the preview window, you may also clearly see the 15 alignment points formed by the 5 horizontal and 3 vertical lines in gray color, where the current default will be indicated with a tiny blue circle.

### Perform Text Insertion

After setting up the desired text style fonts and control attributes, you may also specify the required text justification manner by picking at the radio-buttons provided in the "Text Insertion" group, and then press the **[Insert]** button to perform the text insertion right away. See also **TEXT** command for text justification rules and text insertion procedure.

Note that, as the default text justification for a new text insertion is determined by the Last Text, not by static system variables, if you want to insert text using the text justification setup in **DDSTYLE**, you must insert the text by pressing the **[Insert]** button from **DDSTYLE**.

### Other Buttons

You may press the **[Help]** button to see the helping text. Press the **[OK]** button to conclude the operation. Copyright © 1994 by Autodesk, Inc. All rights reserved. Autodesk reserves the right to alter product offerings and specifications at any time without notice, and is not responsible for typographical or graphical errors that appear in this document. 01/02/00 TD -0.0013 Tc 0 Tw [C an

# DDTEXT

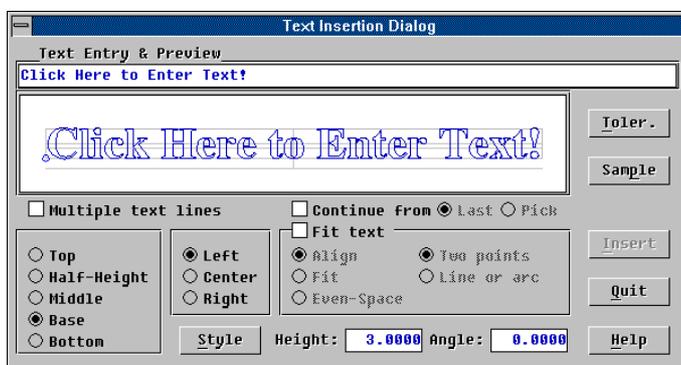
## Dialogue based Text Insertion Command (TCL)

### Purpose:

The **DDTEXT** command is used to insert texts via a GUI dialogue operation manner.

### Description:

The **DDTEXT** command lets you insert a text via a GUI dialogue window operation with an easy control of its text style, justification options, insertion method and so forth. Unlike the original **TEXT** command procedure that requires you to follow the command prompts in a step-by-step manner to create a text, **DDTEXT** expands all the text insertion options in the dialogue window, as shown below, for you to access directly.



You may enter the text to insert from the text entry field and see the text result under current text style setting from the preview window. You may change the text style and edit the text content to your satisfaction, choose the way how the text will be inserted and then click the **[Insert]** button to complete the insertion.

The following describes the details of these screen items from the dialogue windows.

### Text Entry and Preview

The text entry field is where you enter the text to insert. Click it and you will be able to enter new text or edit the current text in the field. The resulted image of the text using current text style will be displayed in the preview window below the text entry field.

The preview window is useful not only in showing the effect of the current text style used for the text creation, but also in justifying the use of the built-in feature text functions and symbols in the text. See the **TEXT** commands for a list of these functions and symbols.

Note that the entry field will initially contain a message string like "Click Here to Enter Text", which will disappear when you click the field to enter text. The same message string will also appear when you empty the text in the entry field.

You may cut all the text from the entry field to the clipboard by pressing <Ctrl/X> key and paste in the text from the clipboard by pressing <Ctrl/V> key. To enter these two control codes to the entry field, you have to press down the Scroll Lock key. An alternative way to enter control codes is to use the ALT-nnn sequence. The <Ctrl/X> can be entered by ALT-24 and <Ctrl/V>, ALT-22.

## The [Toler.] Button

If you are going to create the tolerance string using **DDTEXT**, you may click this button. **DDTEXT** will internally invoke the text input handler **TOLERSTR**, the one called by the **TOLERANCE** command, to handle the text string input, which will help you constructing the tolerance expression using feature text functions.

## The [Sample] Button

You may click this button to sample an existing text entity from the drawing to setup the current text style for the new text creation. You will be asked to pick up the text by the prompt:

Please select the text to copy style:

If there is no valid text entry in the text entry field, the text read from the sample text entity will also be duplicated in the entry field for your further editing.

## Multiple Text Lines

If you are going to create multiple lines of text, you may place a check mark in the "Multiple text lines" option check box. **DDTEXT** will let you continue to enter texts for successive text line creation after the current text is created. Otherwise, **DDTEXT** will exit automatically when the current text is created.

## Continue Text Line

You may choose to continue the text line from the next line to the Last Text or to a specific text entity. The Last Text is the text created previously or most recently. You may enable this function by checking on the "Continue from" option check box and select the desired option from the radio buttons after it in the same line.

Note that if you choose to continue text from the next line to a specific text entity, you will be asked to select the text to continue at text creation. Also, if this function is enabled, all the current style setting will become ineffective at text creation, since the text will be created using the same style informations read from the text to continue from.

## Text Justification Groups

The text justification options are divided into two separated unnamed groups; one for the vertical options and the other, horizontal options. These options are provided in radio buttons. You may specify the required text justification by selecting the options from the corresponding radio buttons.

If the text justification is effective for the text to create, you will see a tiny circle in the preview window showing the relative position of the insertion point with respect to the text. This point is also called the alignment point of the text.

Note that if you choose to create the text by fitting or by continuing after the Last Text or an existing text entity, this text justification options will become ineffective.

## Text Fitting

You may choose to create the text by fitting it

- Between two points, which will require you to designate the two points at text creation, or
- Along a specific line or arc entity, which will require you to pick up the alignment entity at text creation.

You may also choose to fit the text by

- Simple alignment, which fits the text between the fitting span by varying the text height, or
- Direct fitting, which fits the text by varying the text width while the text height is maintained as the specified, or
- Even-space fitting, which fits the text by adjusting the character space first without changing the text width as well as the text height.

You can make the choice by selecting the corresponding radio buttons from the Fit Text group. You can make your choice become effective by checking on the "Fit Text" option check box. Note that if this option is effective, the text justification options will become ineffective.

### The [Style] Button

You may click this button to enter the **DDSTYLE** to setup the current text style for the text insertion. See **DDSTYLE** command reference for further details.

### The "Height:" Field

This is a value entry field specifying the default height of the text to create. It corresponds the default text height of the current style setup by the **DDSTYLE** command.

Note that if the text is to be created via Align Fitting, the actual text height used will be calculated otherwise by text fitting. See **TEXT** command for more details.

### The "Angle:" Field

This is a value entry field specifying the text angle in units of degrees. Note that if the text is created via fitting, the text angle will be calculated otherwise.

### Creating the Text

After you have completed the text entry, the **[Insert]**, become effective. You may click this button to create the text entity. Depending on the way you have chosen to create the text, **DDTEXT** will close the dialogue and ask you to

- Designate an insert point of the text, or
- Designate two points to fit the text, or
- Select a line or arc entity to fit the text, or
- Select the text entity to continue from, or
- Do nothing further.

After the text is created, **DDTEXT** will resume the dialogue again for the next text to create if multiple text line option is enabled; otherwise, it exits.

### Other Buttons

You may press the **[Help]** button to see the helping text. Press the **[Quit]** button to end **DDTEXT** immediately. Pressing <ESC> key also quits **DDTEXT**.

**Procedure:**

@CMD: **DDTEXT** (*return*)  
...[Dialogue Operation]

**Example:**

**DELAY****Delay Command Script Execution for a Period of Time**

---

**Purpose:**

The **DELAY** command is used to delay the command execution for a given period.

**Description:**

The **DELAY** command is mainly used in a script file to delay the execution of the **TwinCAD** commands for a given period of time.

**Procedure:**

@CMD: **DELAY** (*space bar or return*)

@CMD: **DELAY** Delay time in ms: (*value*)

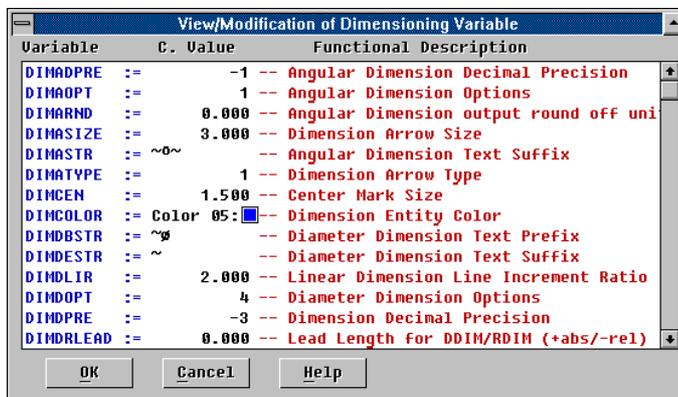
**Example:**

**'DIMVAR****View/Modify Dimension Variables Command****Purpose:**

The **DIMVAR** command is used to access the system variables used for dimensioning.

**Description:**

The **DIMVAR** command lets you access the system variables that control the behavior of dimensioning. It will pop up a variable access window for you to view or to modify these variables. To change a variable from the window, just pick it up by moving the cursor over it and pressing the left most button from the mouse, and **TwinCAD** will let you modify its content.



Note that changing the contents of some of these variables will make all the dimensioning change after regeneration and only after regeneration.

Note also that the ordering of these variables from the window do not follow any sorting rules. They are so arranged subjectively and so believed by the developer that the most frequently accessed variables come first. However, the meaning of each variable from the window is described below in alphabetic order:

**DIMADPRE** Integer, specifies the number of decimal precision for the default text generation of angular dimension. The effect of this variable setting is the same as that of **DIMDPRE** to other type of dimensions, except the following:

- 10 Output degree only, as ddd~°~
- 11 Output degree and minutes, as ddd~° ~mm'
- 12 Output degree, minutes and seconds, as ddd~°~mm'ss"

See also **DIMDPRE** for the effect produced by other value setting.

**DIMAOPT** Integer, bit flag, specifies the default feature options for angular dimensioning:

- Bit 0** *Dimension line option (OD)*, specifying whether to add a circular arc across the angle to measure when the dimension lines (with arrows) are outside of the

angular dimension. If this bit flag is ON, the additional arc will be added joining the two dimension lines at the arrow tips.

**Bit 1** *Text generation option (OT)*, specifying whether to turn the circular text generation off or not. If this bit flag is ON, circular text generation will be turned off, and the text generated for the angular dimension will always be horizontal. If this bit flag is OFF, the text will be in circular fashion around the angle vertex, provided that the **OA** bit is also OFF.

**Bit 2** *Text alignment option (OA)*, specifying whether to align the dimension text with the line drawn from the angle vertex to the text position, in such a manner that the datum dimensioning can be achieved. This bit flag is effective only when the **OT** bit is OFF.

**else** Ignored, but should be zero for future compatible mode.

These default feature options are assigned to the new angular dimension in creation. The interface control of **OD** and **OT** options are also provided when you are dragging the angular dimension (creation by **ADIM** or modification by **CHANGE** command). See also **ADIM** command.

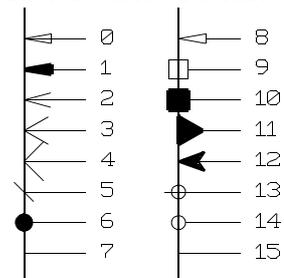
**DIMARND** Real value, specifies the round-off unit for the angular dimension measurement in the default text generation. If it is zero, the round-off function is disabled. For example, to make the default text generation of the angular dimension in units of 5°, specify a value of 5 to this variable.

**DIMASIZE** Real value, specifies the size of the arrow pointer drawn at the end of a dimension line. This is a global variable. Changing this variable will change the size of all dimensioning arrow pointers after drawing regeneration. You can't make the size of an arrow pointer fixed with a particular dimensioning. You can, however, explode the dimensioning so that everything is fixed. Note that the actual size used for a particular dimensioning depends on the scale factor assigned to it.

**DIMASTR** Character string, specifies the unit suffix following the angular dimension measurement in the default text generation. Generally, the degree (°) symbol should be used for the suffix; however, depending on the text style in use, you have to enter the correct code that represents the symbol.

To avoid the code conflict with the extended font style, you may use the '~' (tilde) code to isolate the special symbols.

**DIMATYPE** Integer value, specifies the default type of arrow pointer used for the next subsequent creation of dimension. Valid values are from 0 to 15. See the figure at the right for the type of arrow pointers supported. The arrow type setting will be saved with each dimensioning. You may use **SETDIM** command to alter the arrow type for individual arrow pointer.



**DIMCEN** Real value, controls the drawing of Circle/Arc center marks or center line by the **DDIM** and **RDIM** commands. If it is greater than zero, it specifies the size of the center mark, and the center mark is drawn as well. If it is

less than zero, the center line is drawn besides of the center mark; if it is zero, both are not drawn. This is a global variable. Changing this value will change the marking of dimension center. This is an obsolete variable. You may create permanent center line/mark by **CENDIM** command.

**DIMCOLOR** Color number, specifies the color used for subsequent creation of dimension entities. This value follows the dimension entity.

**DIMDBSTR** Character string, specifies the prefix added before the diametric dimension measurement in the default text generation. Generally, the DIA ( $\phi$ ) symbol is used for the suffix; however, depending on the text style in use, you have to enter the correct code that represents the symbol.

**DIMDESTR** Character string, specifies the suffix following the diametric dimension measurement and the **DIMPOST** in the default text generation. It is an alternative counterpart to the **DIMDBSTR**.

**DIMDLIR** Real value, specifies the default Dimension Line Increment Ratio, which is used to calculate the default increment value of dimension line position based on the height of dimension text. The actual default dimension line increment will be the height of the dimension text multiplied by the dimension scale factor and by this ratio.

For example, if the dimension text size is 5, and a scale factor of 1 is used, with the default **DIMDLIR** value of 2, the actual default dimension line increment will be  $5 \times 1 \times 2 = 10$  drawing units.

**DIMDOPT** Integer, bit flag, specifies the default feature options for diametric dimensioning:

**Bit 0** *Dimension line option (OD)*, specifying whether to generate the dimension line (with arrow) inside of the circle while the dimension text is outside of it. It affects only when the dimension text is placed outside of the circle to dimension. If this bit flag is OFF, the dimension line and arrow will be generated outside pointing to the circle. If it is ON, the dimension line will extend across the diameter of the circle and the arrow pointers are generated as if the dimension text is placed inside of the circle.

**Bit 1** *Text generation option (OT)*, specifying whether to align the dimension text with the dimension line or not, effective only when the dimension text is placed outside of the circle. If this bit flag is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the direction as the dimension line.

**Bit 2** *Text alignment option (OA)*, specifying whether to align the dimension text above the dimension line or not, effective only when the text is placed outside of the circle. If it is OFF, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. If it is ON, the text will be placed above the dimension line (dimension line will be extended to cover the span of text).

Note that this feature option affects only the automatic placement of text when it is in dragging for creation or modification. It is used only to specify the default text position of a dimension entity when it is created or modified interactively. So, you may use SETDIM to move the dimension text to other position.

**Bit 3**      ***Dimension line reverse option (OR)***, specifying whether to reverse the dimension arrow pointer direction when the **OD** option is ON, effective only when **OD** is ON. If this bit flag is ON, and the **OD** option is also ON, then the arrow pointers that were generated inside of the circle will be generated outside pointing into the circle. The dimension line will also be extended over the circle for the additional arrow pointer outside of it.

**else**      Ignored, but should be zero for future compatible mode.

These default feature options are assigned to the new diametric dimension in creation. The interface control of OD, OT, OA and OR options are also provided when you are dragging the diameter dimension (creation by **DDIM** or modification by **CHANGE** command). See also **DDIM** command.

**DIMDPRE**      Integer, specifies the number of decimal precision to output in the default text generation for dimension measurement. If a zero is specified, then the default text generation will contain only the integer part of the measurement. The decimal point will also be removed. If a positive number is specified, then the default text will contain the specified number of digits after the decimal point. If a negative number is specified, the absolute value of the number is also used to specify the number of decimal precision after the decimal point, but the trailing zero of the resulting text string will be removed.

One exception is the number **-1**, which will take 3 decimal precision digits in the default text generation without the trailing zero, for compatible reason.

**DIMDRLEAD**      Double, specifies the default length of the horizontal lead line to the text for the **DDIM** and **RDIM** when the text is dragged outside of the circle and is horizontal. If its setting value is positive and non-zero, it specifies the absolute length value of the lead line in the drawing unit. If the value is negative, it specifies the relative size in ratio to the current arrow size in use. If it is zero, which is default and compatible to the previous setting, it is equivalent to the setting of **-1**, which specifies to use the current arrow size as the default length.

Note that this variable affects only the user interface in the creation and modification of the dimension entity.

**DIMESTYLE**      Extended text style name, specifies the extended text style to use for the dimension text. Default to nothing. This is a global variable. Changing this variable will change the shape of all dimensioning texts using the extended character set.

**DIMEXE**      Real value, specifies the length of the extension line extending over the dimension line. This is a global variable. Changing this variable will affect all the existing dimensioning.

<b>DIMEXO</b>	Real value, specifies the offset amount of the extension line from the dimension points. This is a global variable. Changing this variable will affect all the existing dimensioning.
<b>DIMGAP</b>	Real value, specifies the gap between the dimension text and the dimension line. It affects only when the dimension is dragged for new creation or modification. You may use <b>SETDIM</b> command to modify the dimension text position.
<b>DIMLFAC</b>	Real value, specifies the length factor for the dimension measurement in the default text generation. If it is not zero, all the dimension length measurement will be multiplied with this value for the default text generation.
<b>DIMLOPT</b>	Integer, bit flag, specifies the default feature options for linear dimensioning: <ul style="list-style-type: none"> <li><b>Bit 0</b>      <b>Dimension line option (OD)</b>, specifying whether to add a line across the dimensioning when the dimension lines (with arrows) are outside of the angular dimension. If this bit flag is ON, the additional line will be added joining the two dimension lines at the arrow tips.</li> <li><b>Bit 1</b>      Ignored, but should be zero for future compatible mode.</li> <li><b>Bit 2</b>      <b>Text alignment option (OA)</b>, specifying whether to align the dimension text at the same direction as the extension line in such a manner that the datum dimensioning can be achieved.</li> <li><b>Bit 7</b>      <b>Aligned Measurement Option</b>. This is a global control bit flag affecting the dimension value of a rotated linear dimension only when it is being created or modified by <b>CHANGE</b> command. It controls the determination of dimension value of a rotated <b>ALDIM</b>. If this bit flag is ON, then the measurement value will be taken from the true distance between the two dimension points in despite of the Rotated measurement request. If this bit flag is OFF, which is the default, the measurement value will be the projected value of the distance in the rotated direction.</li> <li><b>Bit 8</b>      <b>Spline Leader Option</b> for Leader Dimension. 1 if for spline leader, and 0 for linear leader. This option controls the behavior of the <b>LEADER</b> command.</li> <li><b>else</b>      Ignored, but should be zero for future compatible mode.</li> </ul> <p>These default feature options are assigned to the new diametric dimension in creation. The interface control of <b>OD</b> option is also provided when you are dragging the diametric dimension (creation by <b>HDIM</b>, <b>VDIM</b>, <b>ALDIM</b> and <b>LDIM</b>, or modification by <b>CHANGE</b> command).</p>
<b>DIMLTOL</b>	Real value, specifies the lower tolerance of the dimension in the subsequent creation of dimension. This tolerance value will be negated before use in the text generation. Note that if both the <b>DIMUTOL</b> and <b>DIMLTOL</b> are equal, the tolerance will be expressed by using the plus-minus sign.

<b>DIMOBLANG</b>	Real value, specifies the oblique angle used in the dimension text generation. This is a global variable. Changing this variable will affect all the existing dimensioning texts. Note that the oblique angle of from the Oblique Dimensioning (from <b>ALDIM</b> ) will override this value.				
<b>DIMOBLARW</b>	Integer, ON/OFF control, specifies whether to generate the oblique feature of arrow pointer used in Dimensioning. This is a global variable, default to OFF for compatible reason.  This oblique feature will generate oblique arrow pointers automatically in the linear dimensionings with oblique specification. This feature will make the open end of an arrow pointer on an oblique dimensioning be aligned with the extension line at the arrow sharp end.				
<b>DIMPFIX</b>	Character string, specifies the prefix added to the default text generated for linear dimensioning and ordinate dimensioning.				
<b>DIMPOST</b>	Character string, specifies general unit suffix following the dimension measurement in the default text generation. It applies to all types of dimensions except the angular dimension.				
<b>DIMRBSTR</b>	Character string, same function as <b>DIMDBSTR</b> , except that it is for Radius dimension.				
<b>DIMRESTR</b>	Character string, same function as <b>DIMDEST</b> , except that it is for Radius dimension.				
<b>DIMRND</b>	Real value, specifies the round-off unit for the dimension measurement in the default text generation. If it is zero, the function is disabled. For example, suppose the round-off unit is 5, then the measurement of a dimension will be rounded off to a multiple of 5.				
<b>DIMROPT</b>	Integer, bit flag, specifies the default feature options for radius dimensioning: <table border="0" style="margin-left: 2em;"> <tr> <td style="vertical-align: top;"><b>Bit 0</b></td> <td><b>Dimension line option (OD)</b>, specifying whether to extend the dimension line (with arrow) across the diameter into the circle while the dimension text is outside of it. It affects only when the dimension text is placed outside of the circle to dimension.  If this bit flag is OFF, the dimension line and arrow will be generated outside pointing to the circle. If it is ON, the dimension line will extend across the diameter into the circle and the arrow pointers are generated at the end of the dimension line pointing to the circle. The effect of <b>OD</b> will be affected by <b>OR</b>. See also <b>OR</b> feature.  This feature option is for dimensioning an arc of small radius, when the text is to be placed "inside" of the arc. See RDIM command for examples.</td> </tr> <tr> <td style="vertical-align: top;"><b>Bit 1</b></td> <td><b>Text generation option (OT)</b>, specifying whether to align the dimension text with the dimension line or not, effective only when the dimension text is placed outside of the circle. If this bit flag is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the direction as the dimension line.</td> </tr> </table>	<b>Bit 0</b>	<b>Dimension line option (OD)</b> , specifying whether to extend the dimension line (with arrow) across the diameter into the circle while the dimension text is outside of it. It affects only when the dimension text is placed outside of the circle to dimension.  If this bit flag is OFF, the dimension line and arrow will be generated outside pointing to the circle. If it is ON, the dimension line will extend across the diameter into the circle and the arrow pointers are generated at the end of the dimension line pointing to the circle. The effect of <b>OD</b> will be affected by <b>OR</b> . See also <b>OR</b> feature.  This feature option is for dimensioning an arc of small radius, when the text is to be placed "inside" of the arc. See RDIM command for examples.	<b>Bit 1</b>	<b>Text generation option (OT)</b> , specifying whether to align the dimension text with the dimension line or not, effective only when the dimension text is placed outside of the circle. If this bit flag is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the direction as the dimension line.
<b>Bit 0</b>	<b>Dimension line option (OD)</b> , specifying whether to extend the dimension line (with arrow) across the diameter into the circle while the dimension text is outside of it. It affects only when the dimension text is placed outside of the circle to dimension.  If this bit flag is OFF, the dimension line and arrow will be generated outside pointing to the circle. If it is ON, the dimension line will extend across the diameter into the circle and the arrow pointers are generated at the end of the dimension line pointing to the circle. The effect of <b>OD</b> will be affected by <b>OR</b> . See also <b>OR</b> feature.  This feature option is for dimensioning an arc of small radius, when the text is to be placed "inside" of the arc. See RDIM command for examples.				
<b>Bit 1</b>	<b>Text generation option (OT)</b> , specifying whether to align the dimension text with the dimension line or not, effective only when the dimension text is placed outside of the circle. If this bit flag is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the direction as the dimension line.				

**Bit 2** *Text alignment option (OA)*, specifying whether to align the dimension text above the dimension line or not, effective only when the text is placed outside of the circle. If it is OFF, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. If it is ON, the text will be placed above the dimension line (dimension line will be extended to cover the span of text).

Note that this feature option affects only the automatic placement of text when it is in dragging for creation or modification. It is used only to specify the default text position of a dimension entity when it is created or modified interactively. So, you may use **SETDIM** to move the dimension text to other position.

**Bit 3** *Dimension line reverse option (OR)*, specifying whether to reverse the dimension arrow pointer direction when the **OD** option is ON, or to extend the dimension line to the center of the circle if the **OD** option is OFF, effective only when the dimension text is placed outside of the circle.

If this bit flag is ON, the dimension line will be extended to the center of the circle, and if the **OD** option is ON, then the arrow pointer will be inside of the circle, and if the **OD** option is OFF, the arrow pointer will be outside. See **RDIM** command for examples.

**else** Ignored, but should be zero for future compatible mode.

These default feature options are assigned to the new diametric dimension in creation. The interface control of OD, OT, OA and OR options are also provided wh-0.2( )-12c(.2(F)-52(O)-eTD 31 Tw [((ov) 28-6.5(f)-1376ln(f)-1376)-6.4( )h)-8

- DIMTADJ** Integer value, specifies the default text adjusting type. A value of **0** specifies that the dimension text is dragged by its start position, i.e., the text is left-adjusted. This is the default condition. A value of **1** specifies that the dimension text is dragged by its middle position, i.e., the text is center-adjusted. It is effective only for linear and angular dimensioning.
- A value of **2** specifies that the dimension text will be automatically placed at the middle of the dimension line, whenever the dragging of the text is inside of the dimension. It is ineffective to the ordinate dimension. All else values are ignored and assumed the value 0.
- DIMTFAC** Real value, specifies the scale factor of the text height in subscript and superscript generation in the dimension text. The subscript and superscript are used for tolerance expression. The default text height of subscript or superscript is half of the normal text height. This scale factor is used to change this default height. The default scale factor is 1. To use full height of the text in the tolerance expression, set the scale factor to 2 or -2.
- The baseline alignment of the superscript and subscript texts can be controlled by the sign of the value. If the value is positive, the text alignment will force the bottom line of the superscript be located at the half-height position of the normal text. If it is negative, the text alignment will use the original baseline for the subscript text and calculate the position of the superscript baseline.
- DIMTOL** Integer value, controls the tolerance expression generation. If it is positive, it enables the tolerance expression and specifies the number of decimal positions the tolerance will be expressed with. If it is 0, then the tolerance expression will be disabled. If it is negative, then the default text will generate the upper limit and lower limit of dimension.
- DIMTRDIST** Real value, specifies the default text row distance used for the dimension text generation, such that the specification of **TXTROWDIST** will not affect the dimension text generation, especially when there are tolerances or limits dimensioning.
- DIMTSIZE** Real value, specifies the height of the dimension text in use. This is a global variable. Changing this variable will change the size of all dimensioning texts after drawing regeneration.
- DIMTSTYLE** Text style name, specifies the text style to use for the dimension text. Default to **STANDARD**. This is a global variable. Changing this variable will change the shape of all dimensioning texts, excepts those being exploded, after drawing regeneration.
- DIMTWIDTH** Real value, specifies the text width factor used in the dimension text generation. This is a global variable. Changing this variable will affect all the existing dimensioning texts.
- DIMUTOL** Real value, specifies the upper tolerance of the dimension in the subsequent creation of dimension.

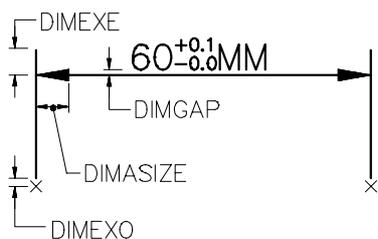
**Procedure:**

Enter the **DIMVAR** command to **TwinCAD** command prompt:

@CMD: **DIMVAR**

and **TwinCAD** will pop up the variable access window. To exit the window, press <ESC> key, or the right most mouse button.

**Example:**



**DIST**

## Measure Distance Between Points Command

---

**Purpose:**

The **DIST** command is used to report the distance between two points.

**Description:**

The **DIST** command reports the 2D distance in drawing units, angle direction and delta-X/Y between two designated points. You may designate the two points in the manner as you draw a line. This is an **AutoCAD**-compatible command.

**Procedure:**

Enter the **DIST** command to **TwinCAD** command prompt:

@CMD: **DIST**

and you will be asked to designate the two points in the sequence below:

Distance from point: (*point1*)

To point: (*point2*)

Len= xxxx, Δx= xxxx, Δy= xxxx, Angle= xxxx°

**Example:**

# DIVIDE

## Divide Objects And Place Marker Command

---

### Purpose:

The **DIVIDE** command is used to divide an object geometrically and place markers on it at each dividing point.

### Description:

The **DIVIDE** command lets you divide a selected object geometrically into parts of equal length, and place markers along it at the dividing points. You will be asked to pick up the object first and then specify the number of parts to divide.

There are several kinds of marks that you can use:

- **Point** - This is the default mark. A point entity is inserted along the object at each dividing point. You can set the system variables **PDSIZE** and **PDMODE** to view the points. See also **SYSVAR** or **POINT** command.
- **Line** - The Line marks are placed normal to the selected object. You may specify the size (length) of the line, and determine how you want to place the line marks relative to the object. The line marks can be positioned on either side of the object, or be bisected by the object in the middle. After pre-setting the line marks, you will be asked to enter the segment length of the measuring unit. The default case for placing the line marks is Middle.
- **Circle** - The Circle marks may be placed tangent to or centered on the selected object. You may specify the size (diameter) of the circle, and determine how you want to place the circle marks relative to the object. In the cases where the circle marks are tangent to the object, you may position the circle on either side of the object. The default case for placing the circle marks is Middle.
- **Block** - The insertion of a given block is made at each measuring point. To start the process, you will be asked to enter the name of the block first, and then to decide whether to align the insertion of the block with the object or not. If the block instance is to be aligned with the object, additional prompt will be given and you may specify a scale factor for the block instance as well as the type of alignment. See Special Note section.

The dividing point starts at the nearest end point from the selected point on the object. For a circle entity, it always starts from the zero angle point and continue in the counter-clockwise direction.

The sub-command options for the mark selections are as below:

- B**      **Block**, request to insert a block mark
- L**      **Line**, request to insert a line mark
- C**      **Circle**, request to insert a circle mark

The sub-command options for the placement of line/circle marks on the selected objects are as below:

- L**      **Leftside**, request to insert the marks on the "left" side of the object
- R**      **Rightside**, request to insert the marks on the "right" side of the object

**M Middle**, request to insert the marks of which the mid-points or the center points fall on the object

Note: The determination on left or right side of the object depends on the direction of traveling along the object. The direction of traveling always starts from the end point which is closer to the picked point, to the other end point.

Only Lines, Arcs, Ellipse, Circles and Polylines are valid for this command operation. See also **MEASURE** command.

## Special Notes

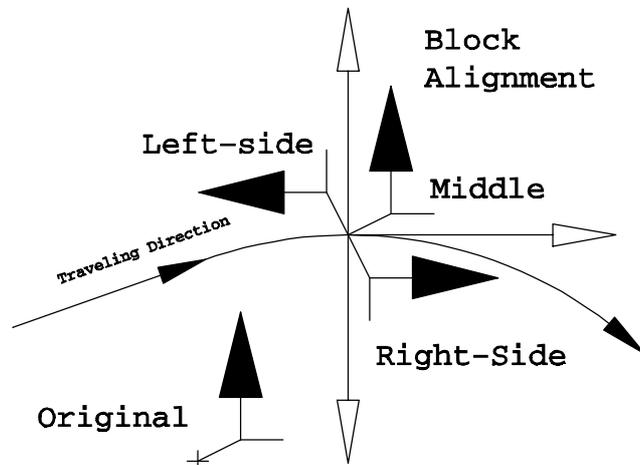
The alignment of block instance with respect to the selected object has been modified since V2.10, which clearly defines the direction of alignment. Now, if the block instance is to be aligned with the object, additional prompt will be given:

Leftside/Rightside/Middle/<Size:1.>:

You may specify the type of alignment by entering the option character "L", "R" or "M", respectively. **TwinCAD** will re-prompt again, and you may specify a scale factor for the block instance to start the operation.

The Left-side option indicates that the alignment vector points to the left side of the object along the measuring direction. The Right-side option indicates that the vector points to the right. And the Middle option, which is the default, aligns the vector with the object's traveling direction. See the figure on the right.

Orientation of Block aligned with Object:



## Procedure:

Enter the **DIVIDE** command to **TwinCAD** command prompt:

@CMD: **DIVIDE**

and **TwinCAD** will prompt

Select object to divide:

for you to pick up the object to divide. See example procedures below.

- **To divide using POINT as marker**

Select object to divide: (*pick one*)

Block/Line/Circle/<Number of segments>: (*value*)

- **To divide using Block Instance as marker**

Select object to divide: *(pick one)*

Block/Line/Circle/<Number of segments>: **B**

Block name to insert: *(select a block from pop-up window)*

Align block with object : **Y**

Leftside/Rightside/Middle/<Size:1.>: **L**

Leftside/Rightside/Middle/<Size:1.>: *(Space bar)*

Number of segments: *(value)*

- **To divide using LINE/CIRCLE as marker**

Select object to divide: *(pick one)*

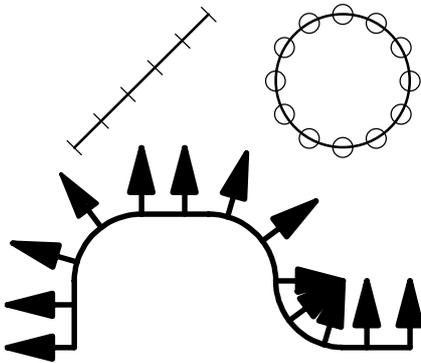
Block/Line/Circle/<Number of segments>: **L**

Leftside/Rightside/Middle/<size (len)>: *(L, R or M)*

Leftside/Rightside/Middle/<size (len)>: *(value)*

Number of segments: *(value)*

**Example:**



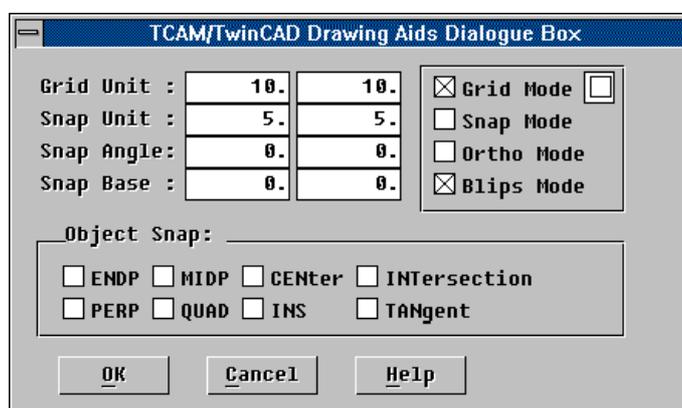
**'DMODE****Drawing Mode Setup Command****Purpose:**

The **DMODE** command is used to set up the current drawing modes in GUI manner.

**Description:**

The **DMODE** command lets you specify the current setup of drawing modes, which include the grid display, cursor snapping functions, marker blip mode, orthogonal drawing mode and object snap running mode, in a dialogue window operation manner.

**TwinCAD** will pop up the dialogue window as shown below:



The dialogue window contains the following value entries:

- Grid Unit** Two entries, specifying the grid units used in the X and Y axis, respectively. The grid units are used to control the spacing of the grid dot when the Grid Mode is turned ON.
- Snap Unit** Two entries, specifying the spacing of the cursor snapping along the X and Y axis, respectively, in cursor Snap Mode.
- Snap Angle** Two entries, the first one specifying the angle of rotation of the first snap axis with respect to the X axis direction in units of degree. The second entry specifies the secondary snap axis direction angle relative to the first axis, and is used to build a non-orthogonal coordinate system in the display. It is effective only when it is not zero or a multiple of 90°. Both angle values will also affect the orientation of the grid dot display and the cross-hair cursor. See also **PUCS** command.
- Snap Base** Two entries, specifying the X and Y coordinates of the cursor snapping origin, which is also the origin of the grid dots in display.

The dialogue window also contains the following option check-boxes:

- Grid Mode** Check to turn ON the grid dot display. See also **GRID** command.
- Snap Mode** Check to turn ON the cursor snapping. See also **SNAP** command.

- Ortho Mode** Check to enable the orthogonal mode. See also **ORTHO** command.
- Blips Mode** Check to turn ON the point blip marking function which generates a point marker (in small crossline) right at the snapped position when you pick up an object or designate a point. The marker thus generated is not a part of the drawing, yet it helps you to identify the pickup operation. It will be removed when you redraw the screen.
- Object Snaps** Check to turn on the specific object snap directives for the general cursor snapping. See **OSNAP** commands.

To confirm the change on these above entries or check-boxes, press the **[OK]** button; or press **[Cancel]** button cancel the modifications.

Note that, if you press <ESC> key or the right mouse button, or pick at the [-] button, **DMODE** will exit as if you press the **[OK]** button to maintain the downward compatibility. If this should trouble you when you try to cancel the modification by pressing the <ESC> key, you may issue **UNDO** command to restore these original setting values.

### Procedure:

Enter the **DMODE** command to **TwinCAD** command prompt:

@CMD: **DMODE**

# DONUT

## Draw Filled Circles and Rings Command (TCL)

---

### Purpose:

The **DONUT** command is used to draw filled circles and rings.

### Description:

The **DONUT** command lets you draw filled circles and rings in a simple way that is compatible to **AutoCAD**'s **DONUT** command. It will prompt in the following sequence:

Inside diameter <nnn>:  
Outside diameter <nnn>:  
Center of doughnut:

The first two prompts ask you to specify the inside and outside diameters of the doughnuts to create. A zero inside diameter value creates filled circles. Negative values are accepted because only their absolute values are taken. However, if the outside diameter is smaller than the inside one, they will be swapped automatically. **DONUT** quits immediately if both of them are zero.

The last prompt ask you to designate the position of the doughnut to create by its center point. It will be repeated after one is created, until you press space bar to end it.

**DONUT** creates circles with wide line property to draw doughnuts.

### Special Notes:

The **DONUT** command is an external command provided by the TCL program file "DONUT.TCL" or "DONUT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **DONUT** command, you may solve the problem by copying the "DONUT.TCL" to the COMMANDS sub-directory.

### Procedure:

```
@CMD: DONUT (return)
Inside diameter <5.>:
Outside diameter <10.>:
Center of doughnut:
...
```

### Example:

**'DOS**

## Enter DOS Shell Command

---

**Purpose:**

The **DOS** command is used to escape **TwinCAD** into a DOS shell.

**Description:**

The **DOS** command will create a DOS shell and let you enter the **DOS** commands (internal or external) as if you were in **DOS**.

Simply enter the **DOS** command to **TwinCAD**'s command prompt, and the **DOS** prompt will come out at the next command line indicating that you are in **DOS** shell. The drawing editor will be temporarily disabled, but all the menus and the function keys are still valid under this **DOS** prompt. You may copy files, list the directories, enter the word processor and other useful package as you would usually do under the DOS.

To exit the temporary **DOS** shell, type <Ctrl/C> or enter nothing but the space bar to the DOS prompt.

**Windows Version Specific**

When you run software programs other than DOS commands, **TwinCAD** will directly shell the program under Windows and wait for its termination. You can not switch the control back to **TwinCAD** to continue any operation until the shelled program has terminated. If you try to operate **TwinCAD**, you will receive the following message:

```
Program Locked by exec("...")
```

where the "... " represent the path name of the shelled program.

If you enter nothing to the DOS prompt but a carriage return, then **TwinCAD** will enter the real DOS, the same one as that provided by the Windows DOS box. You will have to type **EXIT** command to exit the Windows DOS box.

**DOS Version Specific**

When you run software programs other than DOS commands, **TwinCAD** will free up almost all the resources to the DOS and swap itself out for your software program. And, after you finish your job and exit the software programs, **TwinCAD** will resume itself to the point where it left. Nothing will be lost, unless the software program you have just run has left something in the memory. The only limitation is that you can not run the memory resident programs or **TSR** programs (you may run batch file!).

If you enter nothing to the DOS prompt but a carriage return, then **TwinCAD** will swap itself out and actually invoke the DOS's COMMAND.COM. You will have to type **EXIT** command to exit the DOS COMMAND.

**Special Notes**

**TwinCAD** does not release the XMS memory when it is swapped out for the shelled DOS program. If your application needs the XMS memory to run, you may enter a special code command '!' (followed by a space) to the temporary DOS shell to force **TwinCAD** to release the available XMS memory being held for the display list. A simple message as " -- XMS released: nn KB" will be given on the same command line following the '!' command. The

amount of this available XMS memory varies from one system to another system, depending on the total amount of the XMS memory installed.

Since the display list must be clear out to release the XMS memory, after the DOS shell exits, the system will perform a drawing regeneration (REGEN) automatically to rebuild the display list.

**Procedure:**

Enter the **DOS** command to **TwinCAD** command prompt:

```
@CMD: DOS
```

and **TwinCAD** will enter the DOS shell:

```
C:\TCAD>
```

You may enter almost any DOS commands to the prompt, and other DOS applications such as WIN, ACAD, WORD, and so forth.

**'DRAGMODE'****Set Dragging Mode Command**

---

**Purpose:**

The **DRAGMODE** command is used to set the default dragging mode ON, OFF or AUTO.

**Description:**

The **DRAGMODE** command is used to set the default dragging mode. The default dragging mode is **AUTO**. You may turn the dragging mode OFF when you are copying/moving/rotating a complicated part of drawing, so that the operation will not slow down while dragging a large image of drawing. Most of the time, however, the dragging mode should be in **AUTO**.

This is an **AutoCAD**-compatible command.

**Procedure:**

@CMD: **DRAGMODE** *(return)*

@CMD: Set dragmode ON/OFF/Auto <AUTO>: *(ON, OFF, or AUTO)*

**Example:**

## DWGIN

### Load DWG Format Drawing File Command

---

#### Purpose:

The **DWGIN** command is used to load a drawing file in **DWG** format into the current drawing directly.

#### Description:

The **DWGIN** command lets you read in a drawing in **DWG** file format (**AutoCAD**'s local drawing file format) into the current drawing. It will pop up a file window for you to select the file to load in. The rules in reading drawing entities from the DWG file are the same as those described in **DXFIN** command which load in drawing in DXF file format. See also **DXFIN** command for details.

However, the **DWGIN** command will need to read the original SHX file to resolve the name of a SHAPE in reference, such that it can be converted to SYMBOL reference with the correct name, since the DWG file does not contain the name but an index ID. number of the shape.

The **DWGIN** command will try to locate the SHX file from the environment variable **TCADPATH** first, and then the path from variable **AutoCAD**. The shape names are registered when the shape file is referenced for a loading request. Should an SHX file (for shape reference only) not be found from the disk, the **DWGIN** command will simply bypass it, and the symbol name assignment will be in the format as "**SHAPEi\$nn**", where *i* is the shape file ID. that the DWG file has assigned, and the *nn* is the index number of the shape in the shape file.

Note that, like **DXFIN**, before doing DWGIN, all shape files used must have been converted to SMB files by using the **SHX2SMB.EXE** utility. The **DWGIN** command will not convert the SHX to SMB files. It will ask for an existing equivalent SMB file, though it must also access the SHX to resolve the name of the symbol.

#### Special Notes

**DWGIN** can read DWG files from V2.5 to R13. The DWG files from the R14 can also be read in, but not all the 2-D entities from the R14 are supported and converted.

If the drawing read by **DWGIN** command is the very first drawing file loaded explicitly, it will become the current drawing file (same rule for the **LOAD** command).

If the System variable **NOEMPTYBLK** is set to ON, **DWGIN** will remove empty block automatically in reading the drawing file. The block instances that reference to an empty block are also removed.

If the system variable **DWGINPURGE** is set to 1 or 2, the **DWGIN** command will purge off unreferenced BLOCKs from the DWG file during its reading automatically, provided that the drawing file is not in R13/R14 format. If **DWGINPURGE** is set to 1, **DWGIN** will query you for each unreferenced BLOCK before purging it. Note that un-referenced temporary blocks (such as dimension, hatch) will be removed automatically without any confirmation.

#### Procedure:

Enter the **DWGIN** command to **TwinCAD** command prompt:

@CMD: **DWGIN** (*return*)

and **TwinCAD** will pop up a file window for you to select the DWG input file. If a DWG file is properly loaded in, **TwinCAD** will automatically regenerate the drawing to its current extent.

# DWGOUT

## Save Drawing File in DWG format Command

---

### Purpose:

The **DWGOUT** command is used to save the drawing file in either R10 or R12 **DWG** format.

### Description:

The **DWGOUT** command lets you save the current drawing out in **DWG** file format. You may save the drawing in either R10 or R11/R12 format, depending on the current setting value of the system variable **DWGVERSION**. If the content of this variable is 0, **TwinCAD** saves in R10 DWG format; otherwise, it saves in R11/R12 DWG format.

When you enter **DWGOUT** command, you will be asked to specify the name of the drawing to save out via the file window operation. After that, the file is saved and the command ends.

If it is specified by the system variable **PREVIEWOPT** to save a preview image (slide) of the drawing in current view, **DWGOUT** will create the slide file using the same filename as the output DWG file.

The **DWGOUT** command is also called implicitly by **END** and **SAVE** commands to save DWG file when the file name extension is explicitly given as "DWG".

### Drawing Conversion to DWG file

The **DWGOUT** command will save everything possible to the R10 or R11/R12 DWG file so that the **DWGIN** command may retrieve back all the original informations. However, special conversions are required for certain **TwinCAD**'s entities and side-effects have to be noted, as described in the following paragraphs.

#### Solid-Fill POLYLINE

If the polyline contains only 3 or 4 line segments, it will be converted into 2-D SOLID entity, so that **AutoCAD** may generate the required solid-fill effect. Otherwise, it will be output as an ordinary polyline and you won't see the solid fill effect from **AutoCAD**. However, a special flag value will be set to its Group 70 bit codes so that **TwinCAD** will recognize its solid-fill state at **DWGIN**ing. This bit flag value will be preserved by **AutoCAD** as long as you do not use **AutoCAD**'s PEDIT command to de-curve it.

#### Solid-Fill CIRCLE

The solid-fill circle will be converted into close polyline with a proper width specification that is equivalent to the one created by **AutoCAD**'s DONUT command with zero inner diameter. The **DWGIN** command will replace such polyline with a solid-fill circle.

#### Incompatible TEXT

If the text entity is in linear writing with a zero character distance and character is in normal orientation, and the text contains no built-in symbols nor special feature function codes, it will be output directly as an equivalent **AutoCAD**'s Text Entity. Otherwise, it will be torn down into character pieces and grouped as a block of drawing. The block will be named in the format: **\_TXT\$nnnn**, where **nnnn** is its appearing order number.

The text characters will be separated into several Text entities containing each unit of the character (one byte or two bytes, depending on the language encoding). All the other effects produced by the text feature functions and the image of built-in symbols will be exploded into ordinary line/arc segments.

A special **ATTDEF** entity is added to tag with the block. **TwinCAD** will store most of its original information in this tagged **ATTDEF** entity, which is set with Invisible and Preset attribute. If you set the **ATTMODE** to 2 in **AutoCAD**, you will see these tag's content, which is the original text string.

A corresponding **INSERT** is also created in the entity section. As long as the text block is not exploded, the **DWGIN** command will read back its original definition and replace the block/insert with its original text entity.

One side-effect is that the size of the resulted drawing file will be swelled a lot if there are many such text entities that require this conversion. Also, one possible distortion exists in circular text. **TwinCAD** will draw the circular text stroke with a bending effect, while there will be no such effect in the resulted drawing.

The solid-fill state of the TEXT entities will also be saved; however, you won't see the solid fill effect in **AutoCAD**.

## ELLIPSE

As **AutoCAD** does not support the true Ellipse Entities until R13, these will be converted into close polylines of smooth arcs. There is no simple way to hide the ellipse informations into the R10 DWG file so that it can be read back easily by **TwinCAD**. However, as long as the close polyline is left intact after an **AutoCAD**'s editing session (you may **MOVE**, **COPY**, **ROTATE**, **SCALE** and **MIRROR**, but never **PEDIT** nor **TRIM**), the **DWGIN** command will try to recognize it geometrically, and replace it with its original ellipse definitions if it matches.

## Elliptic ARC

The elliptic arc is approximated by an open polyline with smooth arc segments. If the elliptic arc is already a part of a polyline, these smooth arc segments will also be in the same polyline. The elliptic informations will be lost for good. This is the only serious lost of informations after saving to DWG file.

## 360° ARC

As **AutoCAD** does not allow a 360° arc, such arc will be converted into a CIRCLE. It is not possible to restore the original 360° arc from a CIRCLE by the **DWGIN** command.

## The Orientation of ARC

As the DWG file requires an ARC entity be stored with a starting angle and an ending angle in the counter-clockwise orientation, the original orientation of the arc may be lost. That is to say, if the original orientation of an arc is clockwise, it will become counter-clockwise after being saved to DWG file. To most of the CAD applications, the orientation of an arc is irrelevant.

## REGION and Its Associated Hatching

The REGION definition and the entity is converted and stored as Block/Insert data in DWG file. **TwinCAD** will recognizes such entities during **DWGIN**ing. The solid-fill state of a REGION entity is also saved successfully. However, you won't see the solid-fill effect in **AutoCAD**.

The hatching state imposed on a REGION (done by **RHATCH** command) will be expressed with the real hatching lines grouped as a block of drawing, tagged with special **ATTDEF** describing the hatching characteristics.

## DIMENSION

Dimension entities are converted into equivalent DIMENSION entities with the current dimension picture (as viewed in **TwinCAD**) in the dimension block. Additional informations are successfully hidden so that the original definition of these entities can be retrieved back to **TwinCAD** by **DWGIN**.

However, the Auto-Arc Extension features of an **RDIM/DDIM** may be lost if the DWG file has been updated by **AutoCAD**.

Since R10 does not support the Leader, Center mark as formal dimension entities (it merely draws them out as lines), and the Ordinate dimensions are only supported after R11, **TwinCAD** will disguise them as Angular dimension entities in the R10 DWG file. However, as **AutoCAD** will re-produce the dimension picture for a dimension entity when it has been through some certain transformation operations that alter its orientation, like **ROTATE**, the original dimension picture may be changed to something unrecognizable. Nevertheless, **TwinCAD** will reconstruct the correct resulted dimension picture when it load the drawing again.

If you save in R11/R12 format, the Ordinate dimension will be saved in Ordinate dimension of R11/R12 directly.

## SYMBOL

If the SYMBOL image was loaded from an SMB file that was converted from an existing SHX file, the symbol insertion will be converted back to an equivalent SHAPE entity. A loading request of the original SHX file will be added to the Text Style table. In this case, the **DWGOUT** command needs to access the original SHX file to resolve the index code of the SHAPE. If the SHX file can not be found, the conversion will fail, and the symbol will be converted otherwise.

If a SYMBOL image can not be converted to an equivalent SHAPE entity, it will be converted to a Xref block, which is a feature supported by R11/R12. The R10 will read such DWG file that contains block definitions being marked as Xref block, without any offending problem. The user will see the Xref block be displayed as a text insert: "SMB:xxxx", with a text height equal to its Y-scale factor, in R10.

There exists one pit-fall. The R10 will not save the pathname of the Xref file that was stored with the block entity created by **DWGOUT** command. This will make the R11/R12 not able to resolve the Xref due to the missing information, if the DWG file is then submitted to R11/R12. However, **TwinCAD** will read the drawing back without problem, since the pathname is there only for R11/R12.

One possible distortion exists in oblique symbol. Since R11/R12 does not support the oblique transformation over a block of drawing, the Xref block can not show the oblique effect. However, the oblique information will not be lost. **TwinCAD** will successfully retrieve back the oblique angle from the DWG file.

## Incompatible Multiple Block Instances

Incompatible multiple block instances, such as those created by the **RINSERT** command, will be exploded into single block instances and grouped as a block of drawing with a name in the

format: `_INS$nnnn`, where `nnnn` is its appearing order number. A special ATTDEF entity is added to tag with the block to store its original array informations.

## Drawing Level Attribute Tags

Drawing level tag attributes are converted into ATTDEF entities. However, since **TwinCAD** does not require the user to set up the insert point for an invisible attribute tag properly, while **AutoCAD** always display the ATTDEF at the point of insertion, the resulted non-visible ATTDEFs may overlap at (0,0).

## Attribute Tags Attach to Ordinary Entity

The attribute tags that attach to ordinary entities other than block instances will be lost. Such tagging capability is provided by TCL function `add_tag()` and is for special application. The TCL application developers must be aware of this fact and warn their customers about it.

## Linetypes in Output Device Unit

Linetypes will be converted to conform with **AutoCAD**'s standard. The negative unit value of a linetype which specifies to draw the pattern line in device unit will be taken as a value specifying in the drawing unit. However, a special flag will be set to the linetype table entry, so that **TwinCAD** will recognize it and set the negative value again.

## System Variables

System variables that do not have a counter part in **AutoCAD** will be saved as ATTDEFs attaching to a specific block. However, to reduce the overhead, only those variables that have values different from the default will be saved.

## User Variables

User variables are also saved as ATTDEFs attaching to a specific block, so that the **DWGIN** command will read them back.

## PUCS Table Entries

PUCS table entries are also saved as ATTDEFs attaching to a specific block, so that the **DWGIN** command will read them back.

## Tags and Tag Groups

There is no simple way to save the Tags and Tag Groups in the DWG file without introducing too much overhead. **TwinCAD** will save these informations in a TAG file with the same filename. At loading time, the **DWGIN** command will automatically merge in this TAG file if it is present. To suppress the saving of a TAG file at **DWGOUT**, set the bit 15 flag of the system variable **DWGVERSION** to 1.

The separation of the Tags and Tag Groups from the drawing file does not make the drawing file in-complete, since these table objects are not real part of the drawing database. They do not affect anything of the drawing database. They are merely control data for the user interface.

## Entity Line Width

An elementary geometry entity (such as LINE, ARC, CIRCLE, ELLIPSE) with a line width specification will be converted into equivalent POLYLINE with the same line width

specification. The **DWGIN** command will check such a polyline with a width specification and reduce it to elementary geometry entity as possible.

**Procedure:**

Enter the **DWGOUT** command to **TwinCAD** command prompt:

@CMD: DWGOUT *(return)*

and **TwinCAD** will pop up a file window for you to specify the DWG output file.

**Example:**

# DXFIN

## Load DXF Format Drawing File Command

---

### Purpose:

The **DXFIN** command is used to load a drawing file in **DXF** format into the current drawing.

### Description:

The **DXFIN** command lets you read in a drawing in **DXF** file format (**Drawing Interchange File**) into the current drawing. It will pop up a file window for you to select the file to load in.

**TwinCAD** supports most of the entities read from the DXF format; however, the followings need to be noted.

### LINETYPE

The LINETYPE table is loaded and converted to **TwinCAD** linetype specification as possible. See **LINETYPE** command for the linetype specification.

### 3D Entities

All 3D entities, except the 3D-lines, 3DFACE and 2D entities in UCS, are not supported in current release, but will be supported in the future. Note that the height, thickness and UCS of 2D-entities are supported.

### Dimension

Depending on the value of system variable **DXFIOCNV**, the associative dimension (after R11) can be converted in two ways:

- 1. As a block instance of the dimension picture supplied by the DXF file, if **DXFIOCNV** is **0**. This will preserve the dimension images generated by the original DXF's creator, but it will waste drawing space and lose the advantages that the associative dimension entities can bring (e.g. you can not modify the dimension image simply by **CHANGE**), since what you have read in are groups of LINES, ARCs, and TEXTs.
- 2. As an equivalent dimension entity in **TwinCAD** as possible, if **DXFIOCNV** is **1** and there are DIMENSION entities in the DXF file. Associative dimensions as basic entities in DXF file are introduced after R11. They are defined by a set of control points, and a specification of dimension style (variables specifying how the dimension image will be generated).

Note that if there are no DIMENSION entities in the DXF file (such as those produced before R10), turning on **DXFIOCNV** will read in no dimensions, and the original dimension pictures supplied as blocks in the DXF file will be skipped, since it is the DIMENSIONS that are meant to be read.

The conversion of DXF's associative DIMENSIONS into **TwinCAD**'s dimension entities may not be exact, because the two systems make different approaches in controlling the associative dimensions. However, **TwinCAD** will analyze the dimension picture and make the best conversion as possible.

### POLYLINE

Polylines are read and converted by the following rules:

- DONUT will be recognized and converted into solid-filled CIRCLE or CIRCLE with wide line property.
- Whole ellipse will be recognized and converted into true ELLIPSE entity.
- If the polyline has a uniform width, it will be read in as a polyline with wide line property. However, if it contains only a single segment, it will be reduced to the equivalent single entity such as LINE or ARC.
- If the polyline has a variable width, those segments with variable width specifications will be converted into solid areas bounded by closed polylines.
- Otherwise, it is read in without any conversion.

Note that the curve fitting informations are ignored and the redundant vertexes of a polyline will be removed automatically during the reading. A redundant vertex on a polyline may be produced by a line segment of zero length, or an arc segment of zero radius.

## TEXT and STYLE

Text entities are supported. However, as the text fonts used are not with the drawing file, **TwinCAD** will respond in the following sequence when a text font file (**SHX**) loading request is encountered:

1. Report the case to the operator as below:

DXFIN -- Request primary font file: xxxxxx

or

DXFIN -- Request big-font file: xxxxxx

2. Search the local conversion table file "**DXFSTYLE.TBL**" for an equivalent style conversion setup. If an entry is found, the equivalent style will be loaded and go to step 6.

3. Ask you whether to replace it or not as in the prompt below:

Replace it by SHE/SHZ equivalent font file ? <Y>

If you reply YES, go to step 4. Or, go to step 5.

4. Pop up the text font style selection window.

If it is the primary font file in request, then **TwinCAD** will pop up the window for **SHZ** style (small font style); otherwise, it will pop up the one for **SHE** style (extension font style). You can select a replacement style for the request from the window, or load an equivalent **SHZ/SHE** font style from the disk.

If you quit the window, then no equivalence is established, and all the TEXTs that use the style will not be displayed.

Go to step 6.

5. Pop up the file window to load the SHX file and convert it into SHE file if it is a big font.

You will be asked to assign a new style name to it. Note that, if it is a big-font file request, you will be asked to specify the name of the SHE file to convert to, and also the **BSHX.SHD** (shape driver for big SHX file) must be present in the data path specified by the system initial file.

6. Complete the loading of a text style. All subsequent loading of TEXTs that reference to the style will be properly converted into **TwinCAD** text entities. The conversion table file "**DXFSTYLE.TBL**" will be updated if a new conversion relationship is established.

TEXT with empty string will be ignored during the loading operation.

The text may need to be re-justified for Align/Fit/Even-fit alignment upon the new text style assignment (due to the conversion between SHX and SHZ/SHE), since the font matrix varies with different font file.

The text alignment conversion between **AutoCAD** and **TwinCAD** will be such that the Middle alignment of text from R11/R12 is converted to Mid-Height alignment, and the Middle/Center in all versions will be converted to Mid-Height/Center.

If the text font loading request specifies a PFB file (PostScript Type 1 Font) or a SHX small font and the font file can be found from the path specified in **ACAD** or **TCADPATH** environment variable, it will be loaded in directly without asking for text font replacement.

## SOLID and TRACE

The SOLID and TRACE are converted into closed polyline with the color fill attribute.

## BLOCK

All BLOCK entities are loaded in. If there is already a block of the same name in the current drawing, **TwinCAD** will report

Block xxxxx already exist!

and then, after all the loading is done, prompt

Enter prefix string to all newly loaded multiple defined block name:

for you to enter at most two characters to prefix to all the names that are multiple defined.

The anonymous blocks will be loaded and renamed automatically to avoid the duplication of block names. The duplication of block name is likely to happen when the drawing contains Dimension and Hatching blocks. Anonymous blocks are blocks created by Dimensioning, cross-hatching and other applications, used as a way to represent groups of entities by **AutoCAD**. The name of such blocks are irrelevant to the operators. The block names in "**DIMnnnnn**" and "**HATCHnnnnn**" are also recognized as anonymous blocks.

## VIEWPORT

VIEWPORTs are not supported in current release and will be ignored from the DXF file.

## SHAPE

SHAPEs are converted into equivalent symbol references. However, you must have converted your SHX files containing SHAPE definitions into **TwinCAD**'s symbol libraries before reading your DXF file. When the loading of a SHAPE definition file (as SHX) is in request, **TwinCAD** will follow about the same procedure as that for STYLE to ask for an equivalent symbol library. Again, this equivalent relationship is also saved in the file "**DXFSTYLE.TBL**", so that once the relationship is set up, it is set up for good.

## Special Notes

Though the UCS informations of 2D-entities can be read in and the entities in UCS (more correct saying is ECS) can be displayed and plotted by the current release of **TwinCAD**, they are not guaranteed to be correct in current release. UCS operation and related manipulation are not supported explicitly yet in current release.

There are some potential problems arising from the reading of drawing from DXF file:

- **You may lose the exact representation of the original drawing entities.**

Since the DXF file format was designed for simple drawing file interchange between CAD systems and supports only those entities supported by **AutoCAD**, not all kind of possible entities supported by other CAD systems can be exactly representable under the specification of the DXF file format. Equivalence conversion must be carried out and so the original exact representation may be lost. For example, the DXF file does not support ELLIPSE as a basic entity, so the CAD system that does so must convert it into a series of arc or line segments that represent the ellipse in the drawing. And, when they are read in again by the same CAD system, it sees only a series of line and arc segments, not the original ELLIPSE, though the image is about the same. So, the operator can not expect to treat the read in segments nor try to edit them as the original ellipse. So, don't expect to have exactly the same drawing read from the DXF file as the original one you have just DXFOUTed, unless they contain only simple entities such as lines, arcs and circles.

- **You may lose the precision of your drawing entities.**

Since the DXF file employs ASCII texts to represent values in the drawing entities, conversions from binary floating point values to their ASCII equivalents and vice versa, are carried out during DXFIN and DXFOUT operations. Inevitably, such conversions will introduce errors, either of rounding or of truncation, since not all binary floating point values can be exactly expressed by decimal ASCII string, nor can all decimal ASCII strings be exactly represented by floating point binary (e.g., "0.1" can not be exactly represented by binary number). Normally, this precision loss will not cause any problem, because a difference of 0.0000001 may not be perceived by the human reader over the drawing. However, it may cause problems in drawing editing under certain circumstances. For example, a line is supposed to be exactly (in system's internal precision) tangent to a circle, and the loss of precision will lose the tangency relationship between them. Fortunately, **AutoCAD** has devised an equivalent binary format of DXF file. Use of the binary format will avoid such problems. The reading of binary format of DXF file is also supported in current release.

### Procedure:

Enter the **DXFIN** command to **TwinCAD** command prompt:

```
@CMD: DXFIN (return)
```

and **TwinCAD** will pop up a file window for you to select the DXF input file. If a DXF file is properly loaded in, **TwinCAD** will automatically regenerate the drawing to its current extent.

### Example:

# DXFOUT

## Output Drawing in DXF Format Command

---

### Purpose:

The **DXFOUT** command is used to produce selected part or all of the drawing output in DXF file format.

### Description:

The **DXFOUT** command lets you create a DXF file (**Drawing Interchange File**) from the current drawing or from selected objects in the drawing.

**TwinCAD** will pop up a file window for you to specify the output DXF filename. After that, you will be asked to enter the number of decimal place of accuracy to output. Optionally, you may choose to output only selected objects by entering the sub-command option "E" to the prompt, and then select the objects to output. Otherwise, **TwinCAD** will output the whole drawing.

You may choose to output the DXF file in binary format by entering the sub-command option "B" to the first prompt (the prompting message may not have a hint about this). The prompt will be issued again, so that you may choose to enter the sub-command option "E" to output only selected objects. The number of decimal place of accuracy to output is meaningless when the binary format output is enabled, though it may still appear in the prompt.

If only selected objects are output, **TwinCAD** will still create the **TABLE** and the **BLOCK** sections in the DXF file to convey all the necessary informations to make the DXF file complete by itself. The entities will be output in the selected order if and only if the system variable **ENTORDER** is set to 1 prior to the **DXFOUT** command. See also **SELECT** command.

However, not all the entities from **TwinCAD** can be converted to output in DXF file format. Exceptions are described below:

- **DIMENSION**

**TwinCAD** will convert Dimension Image to block of drawing with name in ***DIMnnnnn*** format at DXFOUT. As dimension is a single entity in **TwinCAD**, it can not be directly converted into any equivalent single entity supported in DXF file. During the process, a dimension picture is created in the form of a Block definition, and a block instance of it (INSERT) is also generated.

- **ELLIPSE or ELLIPTIC ARC**

As the DXF file does not support the Ellipse entity, **TwinCAD** will convert the Ellipse entity into an equivalent polyline, which is an approximation of the Ellipse under the effect of the system variable **SPLINESEG** being set to 1. That means, a whole ellipse will be converted to a closed polyline containing 12 arc segments approximating the ellipse.

- **TEXT**

TEXTs are output accordingly, and the text style informations are converted as possible. However, the font files used by **TwinCAD** may not be used by **AutoCAD** or other CAD systems, especially those SHEs that create customized extension fonts. There may be no counterparts in **AutoCAD**'s environment. Nevertheless, these font file loading requests are also included in the DXF file, so that if you have the font file counterparts in SHX format of the same names, the converted output DXF file may be directly loaded in without any modification under **AutoCAD**'s environment. See **STYLE** sub-sections.

Another conversion problem is the uses of the special symbol and Control functions in text string. Some special symbols like degree (°) are explicitly supported by **AutoCAD** in '%x' form. These

will be converted as possible. But some are not. For example, the built-in symbols in escape code sequences, and the text mode control functions are not supported by **AutoCAD**.

**TwinCAD** will try to recognize from the text string the existence of these convertible special codes, and convert them into equivalent '%x' format. However, since **TwinCAD** is multi-lingual and supports extended fonts of different country codes, and hence the interpretation of these character codes depends on the fonts being used, the recognition process will skip the two-byte codes that are greater than A101h.

These two-byte codes will be copied to the output without any checking, provided that the extended codes are effective (extended font is used and enabled for the subsequent characters). The skip of the two-byte codes in conversion checking is intended to avoid the possible mis-conversion of text in Chinese, Korean, and GB-2312 codes.

The state of '~' code will also be monitored during the text conversion output. (The '~' code is used to toggle the effect of extended font, see TEXT command.)

Another problem is the use of non-zero character space. Since the TEXT in DXF file does not support the specification of space between characters, the character space information will not be output, and the DXFIN assumes zero character space for all the TEXT input.

Also, circular TEXTs can not be properly converted to output. They will be output as a linear text in an angle given by starting angle of the circular TEXT. Anyway, you may explode the TEXT entities before output.

The DXFOUT command includes the Text Align/Fit/Even-fit alignment requirement to the output. The Group 72 has been extended to have a value of 6 to specify the Even-Fit alignment, which is not supported but accepted by **AutoCAD** in reading DXF file.

- **STYLE**

The text style controlling in **TwinCAD** is different from that in **AutoCAD**. For each TEXT entity, there can be two style names in use; one for primary font and the other for extended font. Whereas, **AutoCAD** uses only one style name (name of style table) for each TEXT entity, and uses the style name to index to a style table where the extended font file may be specified.

Therefore, the conversion output requires prescanning all the TEXT entities to output, and builds a text style reference table, from which new style names for **AutoCAD**'s text entities are assigned to bind the possible two referenced **TwinCAD** font styles. The tables are output in the TABLE section accordingly, and each output of the TEXT entities will assume the new assigned style name.

The assignment of new style name will be the same as the primary style name for the first output TEXT entity using it. For subsequent TEXT entities referencing to the same primary style name, yet with a different extended style name, a counting index number is appended to the name.

The DXFOUT command will look at the text font conversion table "**DXFSTYLE.TBL**" to determine the equivalent text font used in **AutoCAD** for a specific text font used in **TwinCAD**. If there is no match in the table, the original name of the text font will be used directly.

The above mentioned conversion includes the text for Dimension and Attribute Tags.

- **Multiple Block Instance (MINSERT)**

If the Multiple Block Instance is not a regular one (that is fully compatible with **AutoCAD**'s specification), it will be decomposed into single block instance to output automatically.

- **REGION and Its Associated State Like RHATCH**

REGION is a special entity in **TwinCAD**, and is not convertible to **AutoCAD**-equivalent. Likewise, the state imposed on the region is not convertible too. However, the picture of the RHATCH state of a region will be converted to BLOCK/INSERT with a name in the form of "**HATCHnnnn**". The region entity itself is also converted into BLOCK/INSERT with a prefix of '\_' (underscore) to its region name and with an additional attribute flag of **128**. This conversion is subject to change in future release.

- **Solid Fill**

Solid fill over a closed polyline, a circle or a region, can not be converted to DXF file output, because there is no equivalent function supported in **AutoCAD**. However, a superimposed flag bit value **256** is set to the attribute flag following the polyline entities.

- **SYMBOL**

SYMBOL reference is output as a SHAPE entity.

- **Drawing Level Tag Attributes**

The drawing level tag attributes will be output as ATTDEFs in the Entity Section. The **DXFIN/DWGIN** command will convert such entities into drawing level tag attributes, since they belong to nothing.

## The TCAM/DXF-Extension Format Output

The system variable **TCAMDXFEXT** is provided to control the **DXFOUT** command to produce DXF file with **TCAM/DXF-Extension**. The default is OFF.

If this variable is OFF, the **DXFOUT** command will produce DXF file conforming to the **AutoCAD** specification and should be read in **AutoCAD** without any obvious problem. However, if the drawing contains informations more than the DXF can hold directly, these informations may be lost. For example, Multiple Block Instance in Polar Array will be output in each separate copy of block instance; the polar array informations are lost. Other examples are the character distance and circular text radius which can not be output in standard DXF file; the text in DXF will appear without character distance adjustment and will be in linear writing only.

If this variable is ON, the **DXFOUT** command will produce DXF file with the **TCAM/DXF-Extension**, which extends the DXF format to hold additional geometry informations directly. The DXF file such produced may not be properly read in by **AutoCAD** and by other S/W that read DXF file with strict checking rules. However, these additional informations included in such extension will be very easy to read by applications (such as other TCAM products) that read drawing in DXF format.

## Procedure:

Enter the **DXFOUT** command to **TwinCAD** command prompt:

```
@CMD: DXFOUT (return)
```

and **TwinCAD** will pop up a file window for you to enter the DXF output file. If a DXF file is properly specified, **TwinCAD** will prompt

```
Enter decimal places of accuracy (0-16)/Entities <10.>:
```

for you to specify the output accuracy. You may enter the sub-command option "E" to the prompt to output only selected entities, as the sequence below:

```
Enter decimal places of accuracy (0-16)/Entities <10.>: E
```

```
Select Objects (+): (Do so)
```

```
Enter decimal places of accuracy (0-16)<10.>: (value or space bar)
```

# 'ECOLOR

## Set New Entity Color Command

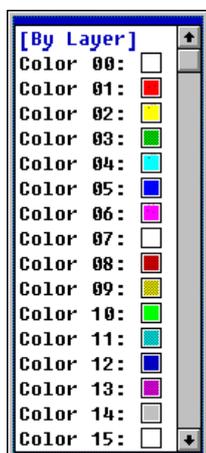
---

### Purpose:

The **ECOLOR** command is used to specify the default color number for new entities of subsequent creation in a GUI manner.

### Description:

The **ECOLOR** command lets you specify the default color for the subsequent new entity creations in a GUI manner. This is a counterpart of the **COLOR** command, because the only difference between them lies in the user interface. **ECOLOR** will pop up a color selection window for you to pick up the color you like.



See also **COLOR** command.

### Procedure:

@CMD: **ECOLOR**  
...[Pick one color]...

**ELEV**

## Set Entity Elevation Command

---

**Purpose:**

The **ELEV** command is used to set the default Elevation and Extrusion Thickness for subsequent new entities in creation.

**Description:**

The **ELEV** command lets you specify the default **Elevation** and **Extrusion Thickness** for the subsequent new entities in creation. The Elevation is the Z plane on which a 2D entity is created, while its extrusion thickness is its height above that base elevation. Negative thickness extrudes downward.

This is an **AutoCAD**-compatible command. You may use **EMODE** command to access both values in dialogue window manner.

**Procedure:**

Enter the **ELEV** command to **TwinCAD** command prompt:

```
@CMD: ELEV
```

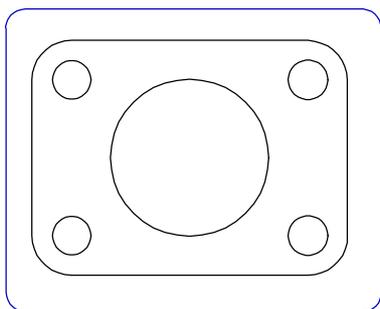
and **TwinCAD** will prompt in sequence as below:

```
New current elevation <0.>: (value)
```

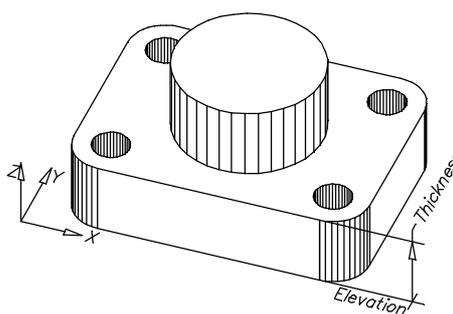
```
New current thickness <0.>: (value)
```

**Example:**

Objects With Elevation  
And Extruded Thickness  
In PLAN View



Same Objects In 3D View



# ELLIPSE

## Draw Ellipse Command

---

### Purpose:

The **ELLIPSE** command is used to draw an ellipse.

### Description:

The **ELLIPSE** command lets you draw ellipses in different ways. The ellipse created by **ELLIPSE** command can be a true Ellipse Entity or a closed polyline comprising of smooth arc segments approximating the curve of the ellipse, depending on the setting of the system variable **ELLIPSEARC**. See later sub-section for further details.

If the ellipses will be approximated with closed polylines, when you enter the **ELLIPSE** command, **TwinCAD** will issue a message like this:

SPLINESEG=1, Total 12 Segments will be produced by estimate.

reporting the fineness of the approximation, which is controlled by the value setting in the system variable **SPLINESEG**. If true Ellipse Entities will be produced, then this message will not be present.

**TwinCAD** will give the first prompt as

Map/Center/<Axis endpoint 1>:

or

Map/Center/Axonometric(ISO)/<Axis endpoint 1>:

depending on whether a PUCS-plane setup is active or not.

You may choose any one of the methods to specify an ellipse by following the command prompts and entering the appropriate sub-command option as it appears.

**M**     **Map**, request to map circles on specific entity planes from a specific view point.

**C**     **Center**, request to specify the center of ellipse.

**A**     **Axonometric**, request to generate ellipses as the projected result of circles under current PUCS-plane setup. This option is present only when the PUCS-plane is active.

The following sub-sections describe these options in details.

### Ellipse by Center/Axis and Eccentricity

In response to the first prompt, you may specify an ellipse by one of its axes and an eccentricity (axis ratio) with respect to it by designating the first endpoint of one axis of the ellipse. You may also define the axes by the center of the ellipse and the end points of the two axes around it, using the sub-command option "**C**".

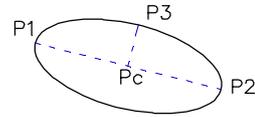
After you have designated the first endpoint of an axis or the center point of the ellipse, **TwinCAD** will prompt

Axis endpoint 2:

asking for the second endpoint of that axis. The axis so defined by the two points may be either the major or minor axis of the ellipse, depending on how you respond to the next prompt:

Rotation/<Other axis distance or '@axis-ratio':>

A dragging of the ellipse is provided with a rubber-band line from the midpoint of the first axis (ellipse center) to the cursor point. The distance between the cursor point and the ellipse center determines the half length of the other axis of the ellipse. You may directly determine the ellipse by



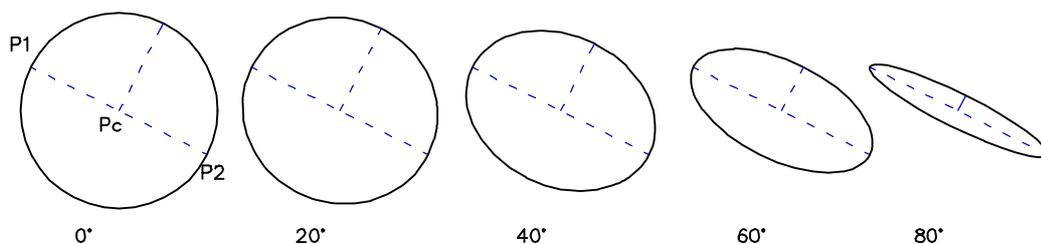
- **Designating a point** which determines the half length of the other axis by its distance from the center, or
- **Entering an absolute distance** which is the half length of the other axis directly, or
- **Entering a relative axis-ratio** in the relative mode format "**@value**" to specify the ratio of the second axis length over the first axis length. (For example, axis-ratio = 1 implies a circle.)

Or, you may enter the sub-command option "**R**" to specify eccentricity by an angle of rotation of a circle with the given axis as its diameter line around which it is rotated. The ellipse is thus determined by the projection of the circle. Upon the sub-command option "**R**", **TwinCAD** prompts

Rotation angle around major axis or '@column-cross-section-angle':

The rubber-band line from the ellipse center to the cursor point, in the dragging of the ellipse, is used to determine the rotation angle. You can drag this cursor point to specify the ellipse dynamically, or enter the absolute rotation angle in units of degree directly to the prompt.

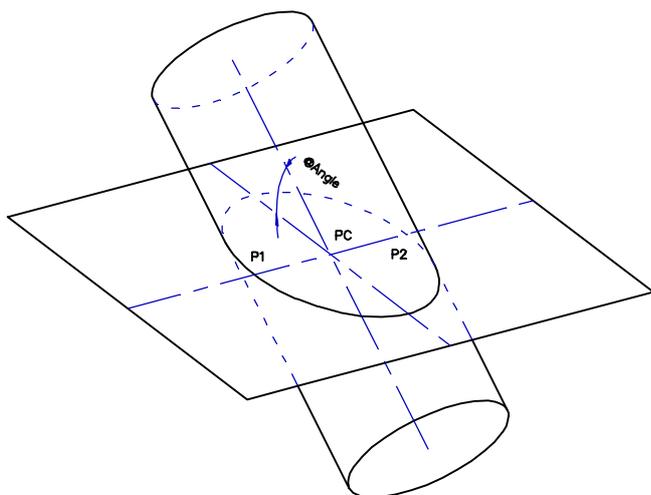
Example of ellipses with varying rotation angle:



### Cross-section of a Column

In specifying the rotation angle of a circle, the relative mode is utilized to specify an ellipse taken from the cross-section of a column by the X-Y plane. The relative value specifies an angle between the center line of the column and the normal vector of the X-Y plane. The ellipse is thus formed as the cross section of the column with the X-Y plane. The given end points of the axis or center point of the ellipse on the X-Y plane indicate the true length of the column's radius/diameter and its slanting direction as well.

Example ellipse drawn as cross-section of a column:



### Ellipse by Mapping View of Circles

You may draw ellipses by mapping circles with given radius and center locations from a specific entity plane with a selected view direction, as if you are using the **MVCOPY** command to generate the ellipses upon these circles, except that you don't need to create these circles in advance. To do so, you may enter the sub-command option "**M**" to the first command prompt issued by **ELLIPSE** command.

You will be asked to set up the mapping conditions (for an Axonometric Projection) by specifying the normal vector of the projection plane (the view direction) and the entity plane where these circles shall reside. **TwinCAD** will prompt in sequence:

```
<Projection Plane (View point:<1.,1.,1.>):  
Top/Right/Front/<Entity Plane:(0.,0.,1.>):
```

After that, you will be continuously asked to designate the center point of the circle (in the model space), and its radius on the virtual entity plane, until you press space bar twice or type **<Ctrl/C>** to exit the operation. **TwinCAD** will prompt in cyclic sequence

```
Center of ellipse:  
Diameter/<Radius:(nn)>:
```

where *nn* is the last radius value for the circle. If you press space bar to the first prompt, the Last Point will be used as the center of the projected circle (ellipse). This is very helpful for drawing concentric ellipses, as the Last Point will be updated by the center of the last created ellipse.

After the center point of the ellipse is given, an ellipse will be dragged dynamically passing through the cursor point as the cursor pointer is moved. You may designate a point through which the ellipse will pass, instead of entering the radius value. The **TAN** object snap directive may be used to specify the ellipse to be tangent to a line (but not a circle nor an ellipse).

If you input a radius or diameter value of the projected circle to define the ellipse, the ellipses so created will be scaled by a factor specified in the system variable **MVSCALE**, provided that it is not zero. You may specify the diameter of the projected circle, using the sub-command option "**D**" to the second prompt.

## Draw Ellipses on PUCS-Plane

If the current PUCS-plane is active, an additional sub-command option "A" will be provided in the prompt as

Map/Center/Axonometric(ISO)/<Axis endpoint 1>:

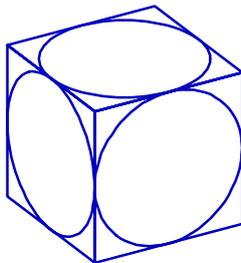
You may enter the sub-command option "A" to draw the ellipse on the PUCS-plane. After that, you will be continuously asked to designate the center point of the circle (in the model space), and its radius on the virtual entity space, until you press space bar twice or type <Ctrl/C> to exit the operation. **TwinCAD** will prompt in cyclic sequencen:

Center of ellipse:

Diameter/<Radius:(nn)>:

The rest operations are the same as those mentioned earlier, except that during the operation, you may alter the current projection axes setup transparently (by **AXOPLANE**, **PUCS**, **ISOPLANE** and <Ctrl/I>, etc.). The change will be immediately recognized and processed correctly, which can be observed from the dragging of ellipse.

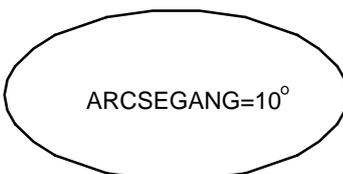
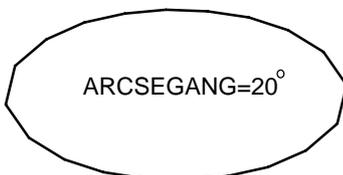
Example ellipses on a Dimetric Cube:



## The True Ellipse Entity

**TwinCAD** supports true Ellipse Entities in the drawing. You may instruct **TwinCAD** to use true Ellipse Entities whenever ellipses or ellipse-arcs are to be created, such as in **ELLIPSE**, **MVCPY** or **EXPLODE** commands, by setting the content of the system variable **ELLIPSEARC** to 1.

True ellipses are drawn to the graphic screen with successive line segments. The fineness of this ellipse curve generation is controlled by the system variable **ARCSEGANG**. An optimum value for **ARCSEGANG** is from 5° to 10°.

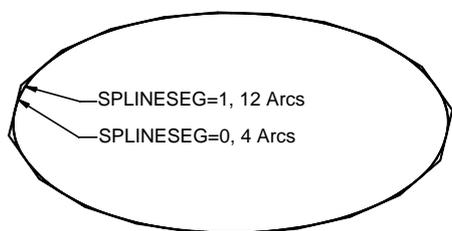


## Approximation of Ellipse Using Arcs

You may explode a true Ellipse Entity and obtain an approximation of it in the form of a polyline comprising of smooth pieces of arc segments. Such an approximation will be automatically generated when an ellipse is created by the **ELLIPSE** command, if the content of the system variable **ELLIPSEARC** is zero.

The number of arc segments used to approximate an ellipse is determined by the system variable **SPLINESEG**, which controls the fineness of the arc approximation of any curve generated. The value of **SPLINESEG** is reported at the beginning of the creation of the ellipse.

If **SPLINESEG** is 0, then only 4 arc segments are used to represent the whole ellipse, and there will be a lot of distortion. If **SPLINESEG** is 1, then there will be 12 arc segments used for the purpose, the result of which is usually quite satisfactory. If **SPLINESEG** is 2, 20 arc segments are used. Add 8 more arc segments to the approximation of an ellipse for every increment of **SPLINESEG**. A value of 1 for **SPLINESEG** is recommended for most of the cases.



See also **SYSVAR** command reference.

### Procedure:

- **To draw ellipses by axis and eccentricity in distance**  
 @CMD: **ELLIPSE** (*return*)  
 Map/Center/<Axis endpoint 1>: (*point1*)  
 Axis endpoint 2: (*point2*)  
 Rotation/<Other axis distance or '@axis-ratio'>: (*point3 or value*)
- **To draw ellipses by axis and relative eccentricity in distance**  
 @CMD: **ELLIPSE** (*return*)  
 Map/Center/<Axis endpoint 1>: (*point1*)  
 Axis endpoint 2: (*point2*)  
 Rotation/<Other axis distance or '@axis-ratio'>: @(axis length ratio)
- **To draw ellipses by axis and eccentricity in rotation angle**  
 @CMD: **ELLIPSE** (*return*)  
 Map/Center/<Axis endpoint 1>: (*point1*)  
 Axis endpoint 2: (*point2*)  
 Rotation/<Other axis distance or '@axis-ratio'>: R  
 Rotation angle around major axis or '@column-cross-section-angle': (*value*)
- **To draw ellipses by center and axis end points**  
 @CMD: **ELLIPSE** (*return*)  
 MAP/Center/<Axis endpoint 1>: C  
 Center of ellipse: (*point*)  
 Axis endpoint: (*point*)  
 Rotation/<Other axis distance or '@axis-ratio'>: (*point3 or value*)

or

Rotation/<Other axis distance or '@axis-ratio':>: **R**

Rotation angle around major axis or '@column-cross-section-angle': *(value)*

- **To draw ellipses by mapping a circle**

@CMD: **ELLIPSE** *(return)*

Map/Center/<Axis endpoint 1>: **M**

Rotate/<Projecting Plane (view point:<1.,1.,1.>)> *(return or value)*

Top/Right/Front/<Entity Plane:(0.,1.,0.)>: **T** or **R** or **F** or *(value)*

Center of ellipse: *(point on entity plane)*

Diameter/<Radius:(*nn*)>: *(return or value)*

...

Center of ellipse: *(space bar twice or <CTRL/C>)*

- **To draw ellipses under PUCS-plane**

@CMD: **ELLIPSE** *(return)*

Map/Center/Axonometric(ISO)/<Axis endpoint 1>: **M**

Center of ellipse: *(point on entity plane)*

Diameter/<Radius:(*nn*)>: *(return or value)*

...

Center of ellipse: *(space bar twice or <CTRL/C>)*

**Example:**

# 'ELTYPE

## Set Entity Linetype Command

---

### Purpose:

The **ELTYPE** command is used to set the default linetype for subsequent new entities in creation.

### Description:

The **ELTYPE** command lets you specify the default linetype for the subsequent new entity creations. It will pop up the linetype selection window for you to select the linetype. The selection will include the "[BY LAYER]" property of linetype.

To quit the selection, depress the right-most button of the pointing device or press the <ESC> key.

This command is a direct access to the Linetype field of **EMODE** command. See also **EMODE** command.

### Procedure:

Enter the **ELTYPE** command to **TwinCAD** command prompt:

@CMD: **ELTYPE**

and **TwinCAD** will pop up the linetype selection window as an example shown below:

LineType	Unit	Descriptions
BORDER	-1.	_____
CENTER	-1.	_____
CONTINUOUS	0.	_____
DASHDOT	-1.	_____
DASHED	-1.	_____
DIVIDE	-1.	_____
DOT	-1.	_____
HIDDEN	-1.	_____
PHANTOM	-1.	_____
* [By Layer]	0.	_____

# 'EMODE

## Set Default Properties for Entity Creation Command

---

### Purpose:

The **EMODE** command is used to set up the default properties for subsequent new entities in creation.

### Description:

The **EMODE** command lets you specify the default Layer, Linetype, Color, Elevation, and Extrusion Thickness for the subsequent new entity creations, in a dialogue window operation manner. See also **ELEV**, **ECOLOR**, **ELTYPE**, **LAYER** commands.

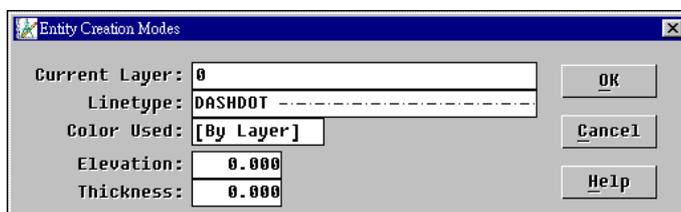
To exit the dialogue window, press <ESC> key from keyboard, or pick the upper left [-] button from the window, or depress the right button of your mouse if you are using one.

### Procedure:

Enter the **EMODE** command to **TwinCAD** command prompt:

```
@CMD: EMODE
```

and **TwinCAD** will pop up the dialogue window as shown below:



<b>END</b>
------------

## Exit TwinCAD Command

---

### Purpose:

The **END** command is used to update the drawing to disk and exit **TwinCAD**.

### Description:

Issue the **END** command, and **TwinCAD** will save the current drawing to disk and then terminate after confirmation. You will have to confirm the action by answering to the following prompt:

Please confirm to save xxxx.WRK and exit? <Y>

You may type 'N' or <Ctrl/C> to cancel the command, if you are not going to leave **TwinCAD**. Otherwise, **TwinCAD** will save the current drawing to disk and then terminate. If this is a new drawing without a name, **TwinCAD** will pop up a file window and require you to enter the name of the drawing to save to.

Note that if the extension of the drawing file name is "DWG", **END** command will save the file in DWG file format by applying the **DWGOUT** command implicitly.

### Windows Specific

If no script is active when this command is issued, a Yes-No-Cancel message box will be pop up for the confirmation.

### Procedure:

Enter the **END** command to **TwinCAD** command prompt:

```
@CMD: END
```

to exit **TwinCAD**.

### Example:

**ERASE**

## Erase Selected Objects Command

---

**Purpose:**

The **ERASE** command is used to erase selected objects from the drawing.

**Description:**

The **ERASE** command lets you remove selected objects from the drawing. You will be asked to select the objects by the object selection operation described in **SELECT** command reference.

You may use the **OOPS** command to restore the last erased objects. However, if possible, it is better to use **UNDO** command instead. This is because, the execution of **OOPS** is not equivalent to the **UNDO** command and must waste some of the space of the undo buffer.

**Procedure:**

To erase objects from the drawing, type **ERASE** to the command prompt

@CMD: **ERASE** (*return*)

and **TwinCAD** will prompt

Select Objects (+):

for you to select these objects to remove. After the object selection is over, the selected objects will be erased and the operation is completed.

**Example:**

# EXPLODE

## Decompose Composite Objects Command

---

### Purpose:

The **EXPLODE** command is used to decompose a composite objects into elementary entities.

### Description:

The **EXPLODE** command lets you decompose composite objects into element entities. A composite object may be a Polyline, Block Instance (**INSERT**), Multiple Block Instance (**MINSERT**,**RINSERT**), Dimension, Region and Text. The decomposition processes for each kind of these composite objects are different and are described in below:

**Block Instance (INSERT)** The explosion of a block instance will cause a copy of these entities that comprise the block to be made at the insertion point. Scaling and rotation are also handled during the copying, such that the image on the screen is identical. If the X/Y scale factors of the block instance is not the same, circles and arc segments will be replaced with ellipses and ellipse-arcs.

Note that all the invisible attribute tags of a block instance will be discarded, as attribute tag can not exist by itself. However, visible attribute tags will be converted into equivalent TEXT entities.

**Multiple Block Instance (MINSERT)** You may choose to explode a multiple block instance into an array of single block instances, or to explode it fully to elementary entities.

**Polyline** The logical links between segments of the polyline are removed. Note that the state of solid fill over a polyline will be removed if it is exploded and the elliptic arc segments will not be further exploded.

**Dimension** Except the dimension text, which is in **TEXT** entity after the explosion, all components are replaced by the individual lines and arcs. Note that solid color fill arrow pointers will be replaced by solid color fill polyline after explosion.

**Text** Texts can be exploded in that it is drawn by lines and arcs. So, the total explosion of texts replaces these lines and arcs with real entities. However, you may choose not to totally explode a text into segments, but to separate it into several text entities containing solely a single character, by setting the value of the system variable **EXPLOTEXT** to 0.

**Attribute Tag** Visible attribute tags will be converted into equivalent Text entities after explosion.

**Region** A region is exploded as a single Block Instance. However, if it has been imposed with a geometry state, the geometry state will receive the explosion first. For example, the **RHATCH** command creates a crosshatch pattern over a region. The explosion of the region will create a real hatching pattern as done by the **HATCH** command, and then remove the **RHATCH** state of the region, and the region remains unexploded. Note that the state of solid fill will be removed if a region is exploded.

The explosion of an **RHATCH** will be limited by the current setting value of **EXPLOLIMIT**. If the number of new entities that will be created after the explosion of an **RHATCH** exceeds this limit, **EXPLODE** will refuse to explode it.

**Ellipse or Ellipse-arc** The explosion of an ellipse (or ellipse-arc) will produce a polyline comprising of smooth arc segments approximating the curve of the ellipse. If it is a whole ellipse exploded, the polyline will be closed. The fineness of the approximation is controlled by the system variable **SPLINESEG**; however, if it is 0, 1 will be used instead.

When you issue the **EXPLODE** command, you will be asked to pick up the object to explode one by one. If there are many objects to be exploded at one time, you may enter the sub-command option "**M**" to signify that. Note that if an **MINSERT** or **RINSERT** is chosen in the Mass-explosion manner, no question will be asked and it will not be fully exploded.

### Procedure:

To explode objects in the drawing, type **EXPLODE** to the command prompt

@CMD: **EXPLODE** (*return*)

and **TwinCAD** will prompt

Select block/region reference, polyline, text or dimension:

for you to pick the object to explode. If you has picked up an object of multiple block instance (**MINSERT**), **TwinCAD** will ask you

Multiple insert selected, fully explode it? <Y>:

whether to explode it fully or not. After the explosion is done, **TwinCAD** will report the result as

Done! -- *nnn* entities exploded.

and continue the prompt

Select block/region reference, polyline, text or dimension:

again, until you press the space bar or type Ctrl/C to quit.

You may enter the sub-command option "**M**" to request a Mass-explosion, as the procedure below:

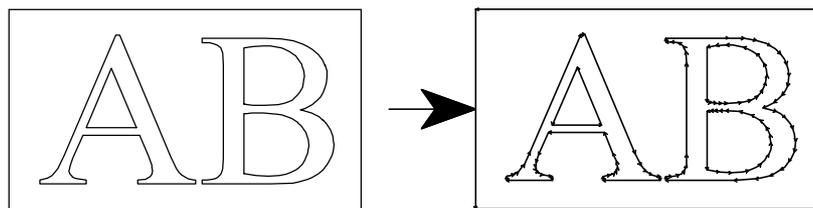
Select block/region reference, polyline, text or dimension: **M**

Select Object (+): (*Do so*)

...

### Example:

#### Explosion of Text



# EXTEND

## Extend the Span of Objects Command

---

### Purpose:

The **EXTEND** command is used to extend the length of objects to given boundaries.

### Description:

The **EXTEND** command lets you change the length of an object by extending its span so that it ends precisely at the nearest boundary selected.

So firstly, you will be asked to select the boundary objects, which may be Lines, Arcs, Circles, Ellipse-arcs and Polylines. See **SELECT** command for object selection operation.

After the boundary objects are selected, you may start extending an object by picking the part of it to be extended. The object you picked will be extended out, from the end nearest to the picked point, reaching to the closest intersection with one of these selected boundaries. If multiple boundaries edges are selected, you may extend an object to successive edges by picking it repeatedly.

You may choose to extend a set of objects by using a series of auxiliary line segments which run across the objects to be extended. By entering the sub-command option "**C**" to the prompt, you will be asked to drag a series of line segments to cross over the objects to be extended. All the objects crossed by the lines will be extended as if it were picked up at the point of the intersection with the crossing lines. This is a convenient way of doing mass extending operation.

You may undo the last modification of an object or objects done by current **EXTEND** command by entering the sub-command option "**U**". To end the command, type <Ctrl/C> or answer the prompt with a null return.

Note that only Lines, Arcs, Ellipse-arcs and open Polylines can be extended in current release. 3D-lines can also be extended as long as they intersect with the boundary entities. Open polylines can only be extended by its ending segments.

A system variable **EXTDEXACT** is used to control whether to extend object to be exactly on the boundary objects in this **EXTEND** command. The default value is OFF, which means that the imaginary geometry of the boundary objects are taken as the boundary. When it is ON, however, the extending operation will only happen when the object can be extended to a point on a selected boundary object.

See also **QCHANGE** and **QTRIM** command for similar operation.

### Update:

For TwinCAD version after V3.1.048, the Text and Block Instance (Insert) entities can also be selected as the boundary objects. The virtual bounding boxes of the texts will be used as the boundary, and the block definitions will be traveled for valid boundary objects.

### Procedure:

- **To extend objects by picking up objects one-by-one**

@CMD: **EXTEND** (*return*)

Select boundary edges....

Select objects (+): *(do so)*

Undo/Cross-line/<Select object to extend>: *(pick object)*

...

Undo/Cross-line/<Select object to extend>: *(space bar)*

- **To extend objects by using Cross-line option**

@CMD: **EXTEND** *(return)*

Select boundary edges....

Select objects (+): *(do so)*

Undo/Cross-line/<Select object to extend>: **C**

First point: *(point 1)*

Second point: *(point 2)*

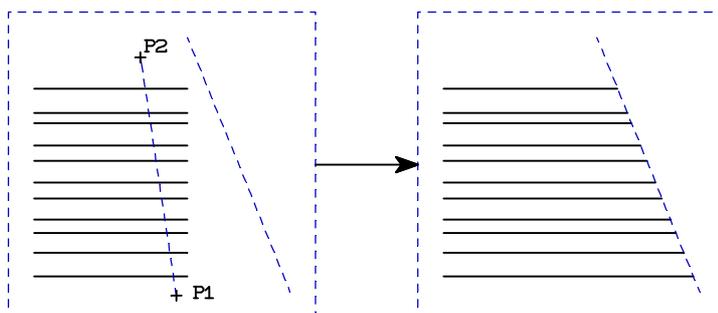
...

Second point: *(point n)*

Second point: *(space bar)*

Undo/Cross-line/<Select object to extend>: *(space bar)*

**Example:**



Extending Objects by Cross-line Option Example

# FILES

## General File Management Command (TCL)

---

### Purpose:

The **FILES** command is used to perform file managements such as file duplication, deletion and renaming.

### Description:

The **FILES** command lets you perform the following file management actions:

- |                           |  |
|---------------------------|--|
| <b>List Files</b>         | Browse the directory files through the file window.  |
| <b>Delete Files</b>       | Browse the directory files through the file window from which you may select one or more files to delete.  |
| <b>Copy Files</b>         | Browse the directory files through the file window from which you may select one or more files to copy. You will be asked later to specify the destination directory through the directory window.   |
| <b>Move Files</b>         | Browse the directory files through the file window from which you may select one or more files to move. You will be asked later to specify the destination directory through the directory window. The files selected are actually copied to the destination directory and then removed. |
| <b>Copy Single File</b>   | Browse the directory files through the file window from which you select one file to copy. You will be asked later to specify the destination pathname of the file to copy to through the file window again.   |
| <b>Rename Single File</b> | Browse the directory files through the file window from which you select one file to rename. You will be asked later to specify the new pathname of the file through the file window again.  |

When you enter **FILES** command, the dialogue window as shown below will pop up.



You may press the corresponding buttons to perform the desired file management action. Press the **[Exit]** button or <ESC> key to end the command.

Note that before doing any real file deletion, copying and moving, **FILES** will ask you to confirm the operation on each selected file. However, you may choose **[ALL]** to confirm all the rest operations.

### Procedure:

```
@CMD: FILES (return)
...[Dialog Operation]...
```

## Fill Solid Color Command

---

### Purpose:

The FILL

**Procedure:**

@CMD: FILL

Select Objects (+): *(do so)*

**Example:**



# FILLET

## Produce Round Corner Command

---

### Purpose:

The **FILLET** command is used to produce round corners between selected objects.

### Description:

The **FILLET** command lets you create an arc with specified radius tangent to two selected objects on the indicated side. The two objects may optionally be trimmed at the tangent points such that the fillet arc segment forms a smooth profile.

Objects that can be selected to make a fillet are Lines, Arcs, Circles and Polylines. You may choose to trim the objects while making a fillet. This is also true for circles. If you choose to trim a circle, the circle will be changed to a 360° arc segment with the start point and the end point at the point of tangency. And later, it may be further trimmed by other operations.

You may choose to make fillets over a polyline such that every corner of it will receive a fillet, automatically. You may change the radius of all the fillets of a polyline by giving a new radius value, and re-fillet it again. Or, you may remove the fillets by supplying a zero radius. The fillet arc segments in a polyline assume a special attribute, so that they can be recognized and modified by the subsequent **FILLET** or **CHAMFER** commands.

When you enter the **FILLET** command, you will be asked continuously to pick up the first object and then the second object to make a fillet, until you respond to the prompt with a null return to end the command. During the operation, several sub-command options are provided:

<value> Direct value taken as radius of the fillet.

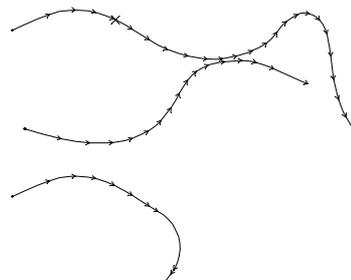
**R** **Radius**, request to change the fillet radius, ACAD-Compatible.

**P** **Polyline**, request to make fillets over a polyline. **TwinCAD** will prompt asking you to select a polyline.

**C** **Circle**, toggle option to break a circle. If it is on, the filleted circle will be converted into an arc with a span of 360°, CCW, so that successive fillets can trim off unwanted portion of the circle. It is effective only when TRIM option is also ON.

**T** **Trim**, toggle option to trim objects. If it is off, the selected objects will not be trimmed. Only a fillet arc segment is produced as required. Note that if TRIM option is on, the fillet arc between the two segments of a polyline will join in the polyline. Otherwise, it will be an independent arc segment.

**J** **Justify**, toggle option to generate proper filleting between curves. When a curve is represented by a polyline which contains successive segments of lines/arcs, and the operator may not properly locate the right segment over which a fillet arc of specific radius can be made correctly without distorting the curve. This option will enable the **FILLET** command to search through the selected curves for the proper fillet arc generation. The



**U**      **Undo**, request to undo the last fillet.

Note that **TwinCAD** supports arcs of 360°. When a circle is first trimmed by **FILLET**, it becomes an arc of 360°, which looks like a circle. So, you probably will mistake such an arc for a circle, and can't figure out why it can't be treated like a circle!

**TwinCAD** allows you to make a fillet over two 3D-lines as long as they intersect at one point (i.e., they lie on the same spatial plane). The resulting arc segment will lie on the plane containing the both 3D-lines. The current release of **TwinCAD** already has the capability to handle such spatial objects in database as well as in display like that; however, there is no such support in current release to help user access or create such objects directly.

## Filleting and Trimming Polylines

Filleting between two polyline segments will trim the polyline(s) at the point of joints, as the rules described below:

1. If the two segments belong to the same polyline, the part of the polyline outside of the fillet arc will be trimmed off. If the polyline is an open polyline and the filleting will make it close, it will be closed.
2. If the two segments belong to two different polylines, the parts of these polylines outside of the fillet arc will be trimmed off. The resulting polylines and the fillet arc will be joined together to form a continuous polyline.
3. If only one segment belongs to a polyline, and the other one is not a polyline segment, then the polyline will be trimmed off by the fillet arc, but no polyline joining will be conducted.
4. The trimming of polylines is controlled by the **Trim** option. Only when this option is ON, will the trimming take place.
5. The determination of the trimmed-off part of the polylines is done so as to make the path smooth from the first segment, along the fillet arc (must be tangent to the first segment), and then joining to the second picked segment. Cusp may be formed at the second joint, depending on various conditions.

## Making Fillets on PUCS-plane

The **FILLET** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to produce the required projected fillet over two lines on the projected plane from the virtual space directly. The result of such elliptic fillet is an Ellipse-arc, despite of the current setting of system variable **ELLIPSEARC**.

The fillet operation will transform the selected objects to the virtual space to make the fillet which is then projected back as the result. The ellipses or ellipse-arcs that can be transformed back to the virtual space as circles may also be selected to make the elliptic fillets.

## Special Notes

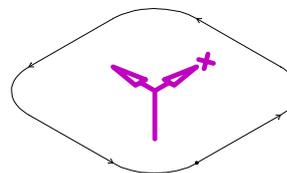
Note that the newly created arc segment of fillet will assume the default properties from the current layer, linetype and color, provided that the fillet is made on two separated entities. It will assume the same properties from the entities that produce the fillet if and only if the entities are two consecutive segments from the same polyline, since the arc segment of fillet will also be included in the polyline.

Also note that a fillet can be produced if and only if the two selected entities lies on the same plane. So, you can make fillet over two 3D-lines, if you are sure that they do intersect in the 3D-space somewhere. But, if the two selected entities do not lie on the same plane, you definitely will receive the error message:

Can't have such fillet.

Common mistakes happens when you are trying to fillet two entities of different elevations.

Another case when you will also receive the above error message, is when the fillet will make either one of the selected entities to disappear. This does not include the case that just makes the entity become zero length (the fillet will be accepted but the zero length entity will be removed). Also note that the Ellipse entity can not be filleted, unless the fillet radius is zero. However, you may explode the Ellipses into polylines and make the fillet with the Justify option.



Elliptic Fillets Generated Under PUCS-Mode by FILLET Command

**Procedure:**

Type **FILLET** command to the prompt

@CMD: **FILLET** (return)

and **TwinCAD** will prompt, as an example, as below:

Undo/Polyline/Radius:<5.>/Circle:<OFF>/Trim:<ON>/Justify:<OFF>/<Select first object>: (pick)

for you to select the first object to fillet. You may enter value to change the fillet radius, or you may enter the sub-command options to toggle the TRIM state or the CIRCLE-BREAK state. After you pick up the first object, **TwinCAD** prompts

Select second object: (pick one)

for you to select the second one. And then, **TwinCAD** continue to prompt

Undo/Polyline/Radius:<5.>/Circle:<OFF>/Trim:<ON>/Justify:<OFF>/<Select first object>: (pick)

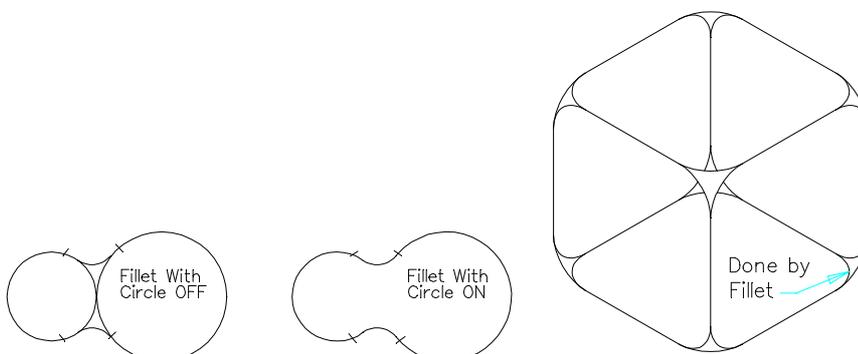
after the fillet is done.

If you enter the sub-command option "P", **TwinCAD** will prompt

Select polyline:

asking you to pick up the polyline to make fillets.

**Example:**



# FNCURVE

## Draw 2-D Parametric Function Curve Command (TCL)

### Purpose:

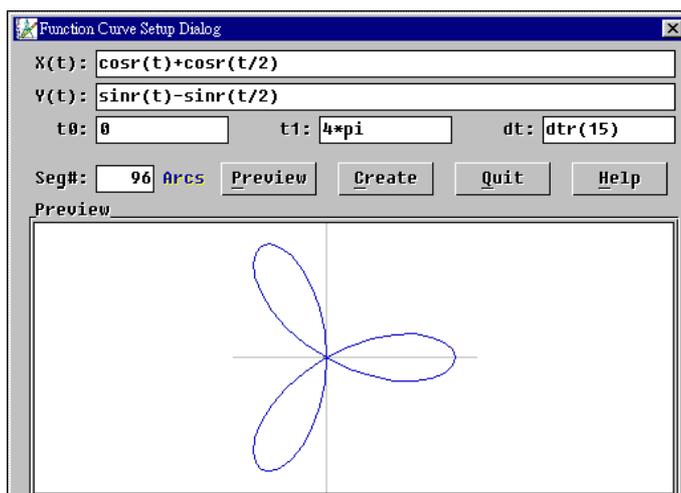
The **FNCURVE** command is used to draw a 2-D plane curve defined by a parametric function.

### Description:

The **FNCURVE** command lets you realize a plane curve defined by a mathematic function in the parametric form:  $P(X(t), Y(t))$ , where  $X(t)$  and  $Y(t)$  are functions that evaluate the X and Y coordinates of a point on the curve at a specific value of  $t$ . The function curve will be realized only in the given range from  $t_0$  to  $t_1$ , where  $P(X(t_0), Y(t_0))$  is the start point and  $P(X(t_1), Y(t_1))$  is the end point of the resulted function curve segment.

### Defining the Function Curve

Upon activation, the **FNCURVE** command will pop up a dialog window as shown in the figure below:



You must supply the correct data to the following text entry fields:

- X(t)** Definition of the X ordinate function, must be given in TCL expression using parameter  $t$  as the only variable.
- Y(t)** Definition of the Y ordinate function, must be given in TCL expression using parameter  $t$  as the only variable, such as the expression " $t*\sinr(t)$ ".
- t0** Start value of parameter  $t$ , can be given in TCL expression, such as " $-2*pi$ ".
- t1** End value of parameter  $t$ , can be given in TCL expression, such as " $-2*pi$ ".
- Dt** Sampling increment of parameter  $t$ , can be given in TCL expression, such as " $dtr(5)$ " which means to evaluate the function points by every 5 °.

### Preview of the Function Curve

Once these above text entry fields have contained correct function curve definition data, you may press the [**Preview**] button to see a preview image of the curve from the preview window.

**FNCURVE** will automatically adjust the graphic extent to cover the whole curve in the given range and draw out the X and Y axis in gray color for reference.

There is a segment counter, beside the [**Preview**] button, which gives the required number of entities to realize the curve under current setting. It will be updated only when the preview image is generated.

## Realization of the Function Curve

You may choose to realize the function curve by

- Creating Point entities on successively evaluated function points, or
- Creating Line segments connecting each successively evaluated function points, or
- Cubic spline fitting over the successively evaluated function points and resulting a smooth polyline containing arc segments.

You may specify the method by picking at the counter unit, which will toggle in the sequence "Pnts", "Lines" and "Arcs" for the above three methods respectively.

If the preview image gives correct result and the required segment counts is also acceptable, you may press the [**Create**] button to create the function curve in the drawing. The function definition data will also be listed out in the command area for latter reference.

Note that **FNCURVE** will always place the function curve's coordinate origin at the drawing's model space origin and the reference axis lines will not be created as seen from the preview image.

## Exit of FNCURVE

**FNCURVE** will exit when you

- Press the [**Create**] button to create the curve and the curve is generated, or
- Press the [**Quit**] button, or
- Press <ESC> key or right mouse button.

**FNCURVE** will save the latest function curve definition data to disk file "FNCURVE.SET" at exit and restore it back at next command execution.

## Special Notes

**FNCURVE** dose not check to see if the TCL expressions for the function definition are syntactically correct or not, nor does it analyze the expressions and give any active warning message about any sort of error. What it does is to pass these expressions to the TCL interpreter for execution. If there is anything wrong, the TCL interpreter will complain it and post appropriate error messages in the command area. You must justify the result by yourself from doing the preview of the function curve before creating it in the drawing.

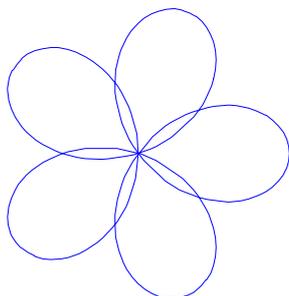
**FNCURVE** does not check the continuity of the function within the given range. It simply evaluates the function points for every increment of the parameter  $t$ . If you choose to generate the curve using line or arc segments, **FNCURVE** will assume the function is continuous in the partition defined by any two successive function points evaluated and make the connection between them. You may need to edit the resulting curve if there should be any discontinuity of the curve in the given range. An example of such function curve is the Tangent function.

The generation of the function curve may be stopped at a point where a mathematic error is encountered in evaluating the TCL expression. It may be a singular point of the function or the value of the parameter  $t$  is out off the valid range. You will have to justify that by yourself.

The **FNCURVE** command is an external command provided by the TCL program file "FNCURVE.TCL" or "FNCURVE.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **FNCURVE** command, you may solve the problem by copying the "FNCURVE.TCL" to the COMMANDS sub-directory.

**Procedure:**

```
@CMD: FNCURVE (return)  
... [Dialog Operation] ...
```

**Example:**

```
Parametric Curve Function:  
X(t) = cosr(t)+sinr(t/4)  
Y(t) = sinr(t)+cosr(t/4)  
T0 = 0.00000, evaluated by '0'  
T1 = 25.13274, evaluated by '8*pi'  
dT = 0.52360, evaluated by 'dtr(30)'
```



# GIFOUT

## Export Drawing Images in GIF File Format Command (Win)

---

### Purpose:

The **GIFOUT** command is used to export the drawing image to disk file in CompuServe Graphic Interchange Format (c) (GIF). The image will be generated in the same way as the **PRPLOT** command prints the drawing to the printer.

### Description:

The **GIFOUT** command lets you export drawing images to disk files in the following GIF image types:

- Monochrome image in white color background.
- Monochrome image in black color background.
- 16-color image in white color background.
- 256-color image in white color background.

The images will be generated in the same way as you prints the drawing to the printer. However, apart from output to the printer, which has fixed resolutions and X/Y aspect ratio, printing image to bitmap file requires you to determine the output resolutions and the X/Y aspect ratio. So, the operation of the **GIFOUT** command is about the same as the **PRPLOT** command, excepts that the details of the image export configuration is different.

In fact, **GIFOUT** command works exactly the same way as **BMPOUT** command, excepts that the resulting bitmap image is converted and saved to the disk file in the required GIF file format.

See **BMPOUT** command for the operation details and special notes about the image file export.

### Special Notes

The image in GIF file format is always compressed with LZW method. According to CompuServe Inc, the following note must be included in the documentation of the program using GIF-files:

```
The Graphics Interchange Format(c) is the Copyright property of CompuServe  
Incorporated.  
GIF(tm) is a Service Mark property of CompuServe Incorporated.
```

### Procedure:

```
@CMD: GIFOUT (return)  
What to plot -- Display, Extents, Limits or Window <D>:  
...[Dialog Window Operation]...
```

### Example:

**'GRID**

## Set up Grid Display Command

---

**Purpose:**

The **GRID** command is used to control the grid display.

**Description:**

The **GRID** command lets you control the state of grid display, and specify the desired spacing between grid of dot to display, by entering a value or sub-command options as described below:

- **(value)** - Spacing between grids in drawing unit in both X and Y directions.
- **ON** - Turn the grid display ON.
- **OFF** - Turn the grid display OFF.
- **S** - Snap, lock the spacing of grid display with the current cursor snapping resolution.

The X-direction of the grid will be the direction specified by the system variable **SNAPANGLE**, and the Y-direction, by the system variable **SNAPBETA**.

Note that the grid display is restricted to the area specified by the system variables, **LIMMAX** and **LIMMIN**. And if the grid is too dense to be clearly displayed in the screen, **TwinCAD** will not show the grid and will give the message:

Grid too dense to display

You may use **DMODE** command to access the same control function in dialogue window manner. See also **DMODE** command.

You can toggle the GRID ON or OFF by typing **Ctrl/G** directly from keyboard.

Use **LIMIT** command to set the size of the grid area. See also **LIMIT** command.

**Procedure:**

@CMD: GRID (*return*)

Grid spacing or ON/OFF/Snap <10.>: (*as required*)

**Example:**

# HATCH

## Create Crosshatch Area Command

---

### Purpose:

The **HATCH** command is used to produce a hatch pattern over a closed area.

### Description:

The **HATCH** command lets you crosshatch a closed area with hatch lines at specified angles and spacing or with predefined pattern. You may select entities with closed boundary, or entities that can be used to define a closed area, for the crosshatching. Valid entities for hatching boundaries are lines, arcs, circles, ellipses, polylines, regions and the virtual bounding box of texts (including tag attributes and dimension texts).

Once you have completed the entity selection, **HATCH** will report:

Total *nnn* single segments selected, working to define boundaries...

and start searching for the possible boundary of closed areas. **HATCH** will call **BPOLY/Auto** command internally to define the closed areas, and show the resulting areas in solid fill color (yellow on black background or blue on white background) with boundaries drawn in dash linetype. See **BPOLY** command for the rules to determine the areas under **Auto** sub-command option.

If no closed area can be defined from those selected objects, the **HATCH** command will prompt

No valid loops found!

and then quit. Otherwise, it will continue to prompt:

Boundary/Pattern/<Base point of hatching line>:

asking you to specify further the hatch pattern specification for these areas. You can do this by

- Applying predefined hatch pattern definition with the sub-command option "**Pattern**", or
- Specifying direct hatching lines,

which will be discussed in later paragraphs.

Once you have completed the hatch pattern specification, **HATCH** will create an anonymous region entity with invisible boundary and with a hatching status in association with the specified hatch pattern to cover these areas. You may modify the hatch pattern specification of such region entities anytime using **CHANGE** or **RHATCH** command. To show its invisible boundary, turn on the system variable **SPLFRAME**.

The closed areas located automatically by **BPOLY/Auto** are areas of maximum coverage by those selected objects. Independent islands will be identified and located as well. However, should you be not satisfied with the areas located in such a manner, you may enter the sub-command option "**Boundary**" to locate them manually. The **HATCH** command will let you define the hatching areas from those selected objects via the general **BPOLY** command operation. See **BPOLY** command for more details.

### Applying Predefined Hatch Pattern

To apply predefined hatch pattern to the closed areas, you must enter the sub-command option "Pattern" to the prompt:

Boundary/Pattern/<Base point of hatching line>:

The **HATCH** command will prompt:

Pattern (? or name) <xxx>:

asking you to specify the name of the pattern, where <xxx> shows the current default pattern name for null return. Note that if the system data register S0 contains a valid pattern name, it will be taken as the current default name of the pattern to use; otherwise, the last pattern name used will be the default.

You may enter a single "?" character to list out all the names of predefined patterns currently loaded in **TwinCAD**. The **HATCH** command will repeat the prompt again after the listing. The listing is very helpful, since each pattern name from the listing is assigned with a sequence number, and you may enter this number to specify it instead of giving its name.

The **HATCH** command will continue to prompt in the following sequence:

Pattern alignment point:

Size/<Scale factor for pattern (1.)>:

Angle for pattern <0. >:

The first prompt asks you to designate an alignment point through which the pattern origin will pass. The default will be at point (0,0).

The next prompt asks you to specify a scale factor of the pattern. You must enter a positive value. If you can not determine a 'proper' scale factor for the specified pattern, you may enter the sub-command option "Size" to calculate it in other way. In response to this option, you will be asked to give a size value of the pattern with the prompt:

Minimum Y-interval for pattern (nnn):

where (nnn) is the size value at current default scale factor. The basic size of a pattern is defined by the minimum Y-interval of hatch lines in the pattern definition. For example, the pattern ANSI32 for steel material has a minimum Y-interval of 0.375 from the original **AutoCAD** definition. It means, if the pattern is generated with a scale factor of 1, the pitch of the repeated pattern of slant lines will be 0.375, which is quite small if you are using the metric system in the drawing. With the "Size" option, you may now directly specify this pitch value.

Finally, you will be asked with the last prompt to specify the orientation of the pattern in unit of degrees. The default is zero degree.

## Specifying Direct Hatching Line

If you answer to the prompt

Boundary/Pattern/<Base point of hatching line>:

with a point or a null return for the default point (0,0), you are going to specify the hatch pattern by direct hatching lines. You will be asked to enter the specification of the hatch lines, including the starting base point, the inclined angle of the lines and the hatch distance between lines. The base point of the hatch lines is the point that one of the hatch lines must pass through. This helps you to align the hatch pattern. The default base point is at the origin.

The **HATCH** command prompts repeatedly in a cyclic sequence:

<Base point of hatching line>:

<Hatching direction/Angle>:

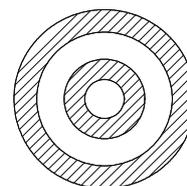
Hatching Distance:

To end the hatch line specification, you must press space bar to the prompt asking for angle or distance. However, there are at most three such hatching line specifications and **HATCH** will end after the third one is specified.

## The Even-Odd Rule Hatching

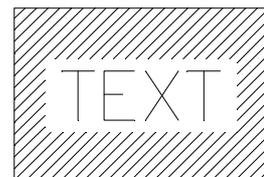
The **HATCH** command always generates the crosshatching lines based on the Even-Odd Rule. This rule determines the "insideness" of a point by drawing a line from that point in any direction and counting the number of boundary segments that the line crosses. If this number is odd, the point is inside; if even, the point is outside.

To generate the crosshatching lines, the crosshatching boundaries are ray-traced by straight lines. The intersection points with the boundaries along each ray-tracing line produce line segments across the crosshatching boundaries. Only those line segments that contains "inside" points will be kept as the crosshatching lines. The Even-Odd Rule is then applied to make the choice. The figure to the right shows the effects of applying this rule.



## Text Virtual Bounding Box in Crosshatching

You may include texts in the selection list for setting up the crosshatching boundaries. A text virtual bounding box will be created for each selected text and used directly for the setup of the boundaries. This also include the texts generated by the Dimension entities.



A gap exists between the text and its virtual bounding box. The gap is assigned a default value of 1/4 of the text height, and may be adjusted using the system variable **TEXTBOXGAP**. When the value of **TEXTBOXGAP** is set positive, the value specifies the gap distance in drawing unit; when the value is set negative, its absolute value specifies the ratio of the gap distance over the text height. A zero value of **TEXTBOXGAP** indicates no gaps at all, and the virtual bounding box will start from the baseline to the text height.

Note that the inclusion of a text does not imply that it will not be crosshatched by the **HATCH** command. It is the Even-Odd rule that determines crosshatching lines.

## The AUTOREGION system variable

The system variable **AUTOREGION** is an on-off control variable, and is used to control whether the **HATCH** command will create an anonymous region entity to generate the direct hatching lines in an associative manner, or to produce the hatching lines in the old compatible manner (hatching lines in a block definition). If it is ON, which is the default, a region entity will be created.

Note that the **HATCH/Pattern** will always create region entity, regardless of this variable setting.

## Special Notes

The rule to find a hatching area by the **HATCH** command in previous version (prior to V3) is quite complicated and unusual to the operator. This is because the searching algorithm has excluded the entity geometry consideration during the traveling of the loops (or connected networks), and the resulted areas may be random for a complicated networks. However, it was fast and quite successful in simple area determination and compatible with the

**AutoCAD's** HATCH command, yet surpassed it in many ways, which was the goal of its original implementation.

Now that the geometry informations are considered during the traveling of the loops (as required by the implementation of the **BPOLY** command), the searching algorithm are thus further improved to actually find out the areas of maximal coverage.

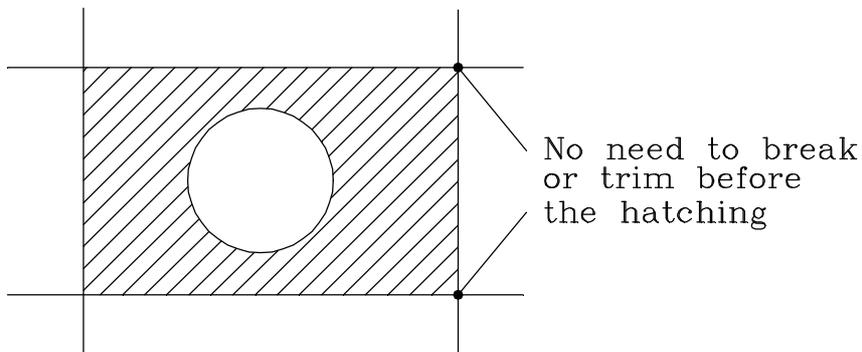
Note that, under the **BPOLY** operation, all close primitives (such as Circle, Ellipse, Close Polyline, Region) will be decomposed into simple line/arc segments for further analysis. They are no longer taken as a whole! This can be a major incompatibility with the version prior to V3.

### Procedure:

To generate crosshatching lines, follow the procedure below:

```
@CMD: HATCH (return)
Select Objects (+): (do so)
Total nnn single segments selected, working to define boundaries...
Boundary/Pattern/<Base point of hatching line>: (point or space bar)
<Hatch direction/Angle>: (value)
Hatch distance: (value)
(TwinCAD generates hatch lines)
<Base point of hatch line>: (space bar)
<Hatch direction/Angle>: (space bar)
```

### Example:



# HATCHEDIT

## Edit Associate Hatch Command (TCL)

### Purpose:

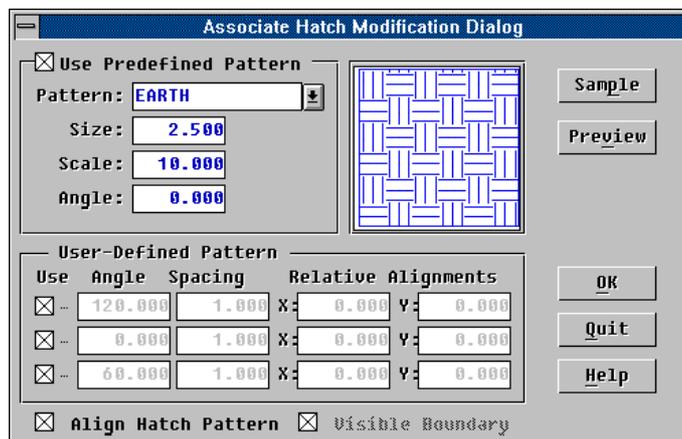
The **HATCHEDIT** command is used to modify the associate hatch state of a selected region entity via a GUI dialogue operation manner.

### Description:

The **HATCHEDIT** command lets you modify the associate hatch state of a selected region entity. You will be asked to pick up the region to change associate state by the prompt:

Select the region to modify its associate hatch:

And then, **HATCHEDIT** will pop up a dialogue window, as shown below, about the same as that of **DDHATCH**.



In fact, **HATCHEDIT** will internally call **DDHATCH** to handle the modification of the associate hatch state of a region entity. See also **DDHATCH** for further operation details.

### Special Notes:

The **HATCHEDIT** command is an external command provided by the TCL program file "HATCHEDIT.TCL" or "HATCHEDIT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **HATCHEDIT** command, you may resolve the problem by copying the "HATCHEDIT.TCL" to the COMMANDS sub-directory.

### Procedure:

@CMD: **HATCHEDIT** (return)

Select the region to modify its associate hatch: (Do so)

...[Dialogue Operation]

### Example:

## HDIM

# Horizontal Linear Dimensioning Command

---

### Purpose:

The **HDIM** command is used to create a linear dimension in horizontal direction.

### Description:

The **HDIM** command lets you generate a linear dimensioning which is always in horizontal direction by

- **Two points**, of which the horizontal projection distance is dimensioned, specifying the extension line origins. If this horizontal projection distance is zero, **TwinCAD** will make a beep and ignore the dimensioning request.
- **A line**, of which the length projecting on X-axis is dimensioned, and of which the two end points are used as the extension line origins. If the line selected is a vertical line, **TwinCAD** will make a beep and ignore the dimensioning request.
- **An arc**, of which the chord length between two end points projecting on the X-axis is dimensioned horizontally. Again, if the length of this projection is zero, **TwinCAD** will make a beep and ignore the dimensioning request.
- **A circle**, of which the diameter is dimensioned horizontally. The default dimension text will be generated as that for the Diametric dimensioning.

You will be asked to designate the first dimension point, and then the second dimension point. If you give the first prompt a null reply, you will then be asked to pick up object (Line/Arc/Circle) to dimension.

The dimension text is generated automatically in the drawing unit to reflect the true length of the object. Dragging of the dimensioning result is provided and is started for you to settle its placement, as soon as you finish the basic specification of the dimensioning.

There are several ways to settle the placement of dimension text as well as the dimension line. During the dragging of dimensioning, you may

- **Enter a distance value**, specifying the distance from the dimension line to the first dimension point, where the dimension text position is determined automatically by the current dragged position when the value is entered. Note that the dragged position determines only the projected position of the text to the dimension line.
- **Enter nothing but space bar**, specifying a default distance for the dimension line should be used. The determination of a default distance depends on the mode of dimensioning. See later explanation.
- **Designate a point**, specifying directly where the dimension text should be placed at. The dimension line will be generated automatically according to the setting of dimension variables (specifically, **DIMGAP**). You may use the cursor snap function (in small step) to help you locate the desired dimension line position, if casual alignment with other dimension lines is intended.

Before settling down the placement of the dimensioning, you may modify the dimensioning by the following sub-command options:

**D** ***Default text***, specifying to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about it.

**C** ***Change text***, specifying to change the dimension text manually. **TwinCAD** will prompt:

Dimension Text:

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

**OD** ***Dimension-Line Option***, specifying to toggle the current effect of dimension line option. The dimension line option is effective only when the dimension text is dragged outside of the dimension points and so are the dimension lines generated outside of the extension lines of the linear dimensioning. When this option is ON, an additional straight line will be added inside of the extension lines, connecting the two dimension lines which fall outside of the extension lines. This option can be preset at the bit 0 of the dimension variable **DIMLOPT**.

**OT** ***Text generation option***, specifying to toggle the current effect of text generation option. It specifies whether to align the dimension text with the dimension line or not. If this option is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the same direction as the dimension line. This option can be preset at bit 1 of the dimension variable **DIMLOPT**.

**OA** ***Text alignment option***, specifying to toggle the current effect of text alignment option. It specifies whether to align the dimension text above the dimension line or not. If this option is ON, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. The dimension line will be clipped if necessary. If it is OFF, the text will be placed above the dimension line (and the dimension line will be extended to cover the span of text if necessary). This option can be preset at bit 2 of the dimension variable **DIMLOPT**.

**OR** ***Dimension-Line Reverse Option***, specifying to toggle the current effect of dimension line reverse option, which means to reverse the dimension line with respect to the dimension text. If this option is ON, the dimension line and arrows will be placed in the reverse direction. That is, if the text is inside, the arrows will be outside pointing inward. If the text is outside, the dimension line will be inside the extension lines. This option can be preset at the bit 3 of the dimension variable **DIMLOPT**.

**A** ***Align Option***, specifying to settle the dimension line placement, such that it can be fixed to align with a specific point or at a specific dimension distance. **TwinCAD** will prompt:

Indicate a point to align the dimension line:

asking you to designate the alignment point. You may directly pick at an existing dimension to align with it, as **TwinCAD** will automatically issue the **ENDp** snap directive, or enter a specific distance value, to fix the dimension line position. Once the dimension line position is fixed, only the dimension text position can be dragged and changed by the cursor pointer.

You may release the fixed dimension line by entering this option again and pressing space bar to the prompt.

After the first dimensioning is done, **TwinCAD** will continue to prompt for the same dimensioning operation with the following added options:

- U**     **Undo**, request to undo the last dimensioning.
- B**     **Base**, request to generate Base Dimensioning, using the first extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimension set.
- When the Base dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Base dimension mode may be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.
- C**     **Continuous**, request to generate Continuous Dimensioning, using the second extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimensioning.
- When the Continuous dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Continuous dimension mode may also be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.
- Note that once the Continuous or Base dimension mode is entered, it will be continued automatically after each dimensioning is done. However, you may press the space bar to the prompt asking for the second dimension point to quit the mode and to go back to the first prompt where you may have other options as usual.
- H**     **Horizontal**, request to switch to horizontal linear dimensioning which is meaningless in this context as you are doing horizontal linear dimensioning.
- V**     **Vertical**, request to switch to vertical linear dimensioning as if you have exited the operation and enter the **VDIM** command.
- A**     **Aligned**, request to switch to aligned linear dimensioning as if you have exited the operation and enter the **ALDIM** command.

The last three sub-command options from the above are added for easy mode switching between the Vertical, Horizontal and Aligned Linear Dimensioning. You may identify which mode you are in now by checking the parenthesized character (**H**, **V** and **A**, respectively) from the command prompt.

You have to type <Ctrl/C> to exit the command operation or give a null reply when you are asked to pick up object to put dimension.

## Default Dimension Line Distance

The determination of default distance of dimension line follows the rules below:

- If Base dimension mode is selected, the default distance will be the last dimension line distance plus or minus the default dimension line interval, depending on whether the second dimension point is outside of the extension lines of the last dimension or not.
- If Continuous dimension mode is selected, the default distance will be determined such that the dimension line of the newly created one will be on the same horizontal level with that of the last one.
- If none of the above situations happens, the default distance will be the default dimension line interval specified by the dimension variable **DIMDLIR** (see **DIMVAR** for details).

## Suppression of Extension Lines

You may use the dimension variables **DIMSE1** and **DIMSE2** to specify the default suppression status of dimension extension lines. However, it is advised not to do so, since it will slow down your productivity. It is better to use the **SETDIM** command to adjust these items after all the dimensioning jobs are done. Do not get annoyed with these variables during the dimensioning operations. They are provided for other purposes, such as dimensions generated by TCL programs. The **SETDIM** command is so designed as to make you feel free with the dimensioning operations.

However, for Base Dimensioning and Continuous Dimensioning, it is obvious that some extension lines would overlap as they are generated. It is therefore necessary to suppress one of them (the shorter one, of course). Sophisticated check is performed automatically to see which one should be suppressed when the dimensioning is done under these two modes. The effects of the **DIMSE1** and **DIMSE2** are ignored in these two modes.

## Virtual Segments Extension

The virtual segments generated by Dimension or Symbol entities can be accepted as valid object selections during Dimension operations. For example, you may select the extension lines and put on dimensions using the **HDIM** command.

## Dimensioning on PUCS-Plane

The **HDIM** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to produce dimensions on the projected plane from the virtual space directly.

If the current PUCS-plane is active and valid for an Axonometric Projection, the dimension points will be measured from the Projected UCS-plane, and the **HDIM** command will automatically calculate the required Oblique angle and set up the Rotated direction in the Y-axis direction for the dimensioning as a projection result of the horizontal linear dimensioning from the Projected UCS-plane to the model space (as the same transformation result done by **MVCOPY** command).

The Ellipse and Elliptic-arcs that can be produced by using current Axonometric Projection setup can be picked up for the linear dimensioning. The result of the dimensioning will be the same as the dimensioning of a circle or an arc on the projected plane from the virtual space.

The **HDIM** is actually a Rotated **ALDIMs** with zero oblique angle, since their extension lines are always fixed to the Y-axis (vertical) direction.

## Procedure:

Enter the **HDIM** command to **TwinCAD** command prompt:

@CMD: **HDIM** *(return)*

and **TwinCAD** will prompt

Horizontal/Vertical/Aligned/(H)<First dimension point>:

If you reply to it by designating a point, you are dimensioning by two points. If you reply to it by pressing the space bar, **TwinCAD** will let you dimension by selecting objects. See the procedure for each case below:

- **Two points:**

Horizontal/Vertical/Aligned/(H)<First dimension point>: *(point)*

Second dimension point: *(point)*

Default-text/Change:<text>/OD(Dimension-line option)/

<Set dimension line position or distance>: *(Drag to position)*

- **A line, arc or circle:**

Horizontal/Vertical/Aligned/(H)<First dimension point>: *(space bar)*

Select arc, line or circle: *(pick up one)*

Default-text/Change:<text>/OD(Dimension-line option)/

<Set dimension line position or distance>: *(Drag to position)*

After the first dimensioning is done, **TwinCAD** prompts

Horizontal/Vertical/Aligned/Base/Continue/Undo/(H)<First point>:  
to continue the operation.

**Example:**

**'HELP**

## Display On-Line Help Command

---

**Purpose:**

The **HELP** command is used to display on-line help text.

**Description:**

The **HELP** command displays the content of a system help file of which the name is specified in the system initial file. The content of **TwinCAD** help file provides quick informations about the operations and the functions of the commands in **TwinCAD**, and all other informations that might be helpful to you through your operation with **TwinCAD**.

**Windows Version Specific**

**TwinCAD** for Windows will call **WinHelp** to display the content of Help file. The help file will be in Windows standard help file format. The original help files for the DOS version in TCAM help file format are no longer supported in **TwinCAD** for Windows.

Depending on the Windows language version as well as the current **SWITCH** command status, **TwinCAD** will display one of the following help files:

**TWINCAD.HLP** Standard English version, also used under non-switched status.

**CTWINCAD.HLP** Traditional Chinese (BIG5) version, used under switched status.

**BTWINCAD.HLP** Simplified Chinese (GB2312) version, used under switched status.

**JTWINCAD.HLP** Japanese (SJIS) version, used under switched status.

**KTWINCAD.HLP** Korean (KEUC) version, used under switched status.

**TTWINCAD.HLP** Thai version, used under switched status.

If you are going to use a specific HLP file other than the above default, you have to setup the parameter "SystemHelpFile" in either "[Default Language]" and "[Alternate Language]" section from the system initial file.

Unlike the DOS version, you don't need to quit the help to continue **TwinCAD** operation.

**DOS Version Specific**

What you would see from the **HELP** command depends on the content in **TwinCAD** help file. In general, **TwinCAD** will open a text slide window and display the texts of interest. Sometimes, you may select different subjects to read by picking at some screen buttons provided in the window.

The help file is an ASCII text file, containing lines of help informations and commands to control the display of these informations. You may prepare your own help files, or edit the one supplied with the package to add more informations to it. See Appendix for the instruction to customize the help file.

The help file must contain labels to tell **TwinCAD** where to start the display. A label named **COMMANDS** will be located at the start of the display, when the **HELP** command is issued at

the top level of command prompt. If you have already issued a command and **TwinCAD** prompts under the command level, the issue of the **HELP** command (either by entering **'HELP** to the prompt transparently, or by typing **<Ctrl/H>** from the keyboard) will cause **TwinCAD** to start the help display from a specific label (usually the name of that command) in the help file. If the label is not found, **TwinCAD** will start the display from the very beginning of the help file, where a general note is usually placed.

You may quit the help display anytime by pressing the **<ESC>** key from the keyboard, or picking **[-]** from the upper left corner of the text window if a window title is present.

Note that **TwinCAD** provides example help files in different languages, make sure you have the right one specified in the system initial file.

**Procedure:**

- To see what is in the help file, type  
@CMD: **HELP** *(return)*
- To see help file for a particular command  
@CMD: **HDIM** *(return)*  
Horizontal/Vertical/Aligned/(H)<First dimension point>: **'HELP**

**Example:**



## Identify Objects Command

---

### Purpose:

The **ID** command is used to identify the geometry and property of an object.

### Description:

The **ID** command lets you quickly identify the geometry and property of a specific object. You pick up an object, and **TwinCAD** will list out the related information about the object.

You may also identify a specific point by using object snap functions like **ENDp**, **INT**, **MID**, **QUA**, **NEAr**, **CEN**, and so on. In such cases, the **ID** command will report the coordinates of the designated point instead.

To quit the command, type **Ctrl/C** or reply to the prompt with a null return.

The listing generated by **ID** command is the same as that by **LIST** command. The difference is that **LIST** command always lists entities from top level, and nests the listing down one level if applicable (such as **POLYLINE** and its segments). While the **ID** command lists the object from the bottom level, and optionally, traces up to the top level when applicable.

For example, if you pick up a segment of a polyline, the information of that segment will be listed out first, and then the information of the polyline containing that segment is also displayed.

A system variable "**EIDLEVEL**" is used to control the level of listing from this **ID** command. Default value of **EIDLEVEL** is 1 (Bit 0), which makes the **ID** command to list out the mother entity of the picked object when applicable. When **EIDLEVEL** is set to be 0, the listing will contain only the object picked. Additional listing controls may be achieved by setting Bit 1 and Bit 2 values of **EIDLEVEL** as described below:

- Bit 0:** If 1, Enable listing of Mother Entity
- Bit 1:** If 1, Enable listing of Referenced Block Content
- Bit 2:** If 1, Enable listing of Polyline segment in Block

As indicated above, the only way for you to list out the content of a Block definition without exploding the Block will be to enable Bit 1 and Bit 2 of "**EIDLEVEL**" variable before you pick up a block instance (**INSERT**) via **ID** command.

Note: You may set Bit 0, Bit 1 and Bit 2 to "Enable" state by simply input binary codes under the command prompt as follows:

```
@EIDLEVEL=111b
```

where "**b**" stands for binary code.

### Procedure:

To identify an object, follow the procedure below:

```
@CMD: ID (return)
Select object to verify: (pick object)
...
Select object to verify: (space bar)
```

'ID

**Example:**

# INSERT

## Insert a Block Instance Command

---

### Purpose:

The **INSERT** command is used to create a block instance (reference to a block).

### Description:

The **INSERT** command lets you create a block instance, which may be scaled and rotated as required, and inserted at a designated position in the current drawing. The block instance can be made as a reference either to an existing Block in the current drawing or to an external drawing in WRK/DWG format. In the latter case, the external drawing is loaded and grouped as a Block with a specified reference name, before the block instance is made.

When you issue this command, **TwinCAD** will ask you first to enter the Block name to insert by the following prompt:

Block name (or ?) *<blkname>*:

where *blkname* is the Block name of the last reference and will be taken as the default if you reply with a null return.

You specify the Block name by typing it from the keyboard in the format as below:

*{\*}{?}{blkname}{={wrkname}}}{<spacebar or return>*

The portions enclosed with braces are optional. You can see that all of them can be optional except the last one which terminates the input string. The meaning of each portion is described below:

**\*** An optional asterisk, prefixed to the input string, which specifies to explode the block instance after the insertion. If this character is not present, the insertion will be a pure block instance; otherwise, the block instance will be exploded automatically, as if the insertion were a copy of the original block definition instead of a reference to it.

**?** An optional question mark, given immediately following the prompt or after the optional asterisk, tells **TwinCAD** to pop up a slide window for Block name selection. All the defined Block names are in the slide window, so that you don't need to memorize nor to type the name you desired. All the rest of characters before the optional equal sign (=) in the input string are ignored. Note that this is the only way for you to insert a Block with an asterisk character (\*) as the first character in its name, to avoid automatic explosion.

**blkname** Character name string, specifying the name of the block to insert, is checked against those Block names already defined in the drawing. If the specified Block name exists, it will be used as the reference target of the block instance. If it does not exist, **TwinCAD** will search the current log-in disk for the drawing with the same name to load in as a Block, unless otherwise specified by the string after the optional equal sign (=).

Note that if this block name is omitted and an external drawing file name is given after the = sign, the filename will be used as the block name.

- =** An optional equal sign, immediately after the Block name, tells **TwinCAD** to define the Block by an external drawing file before making the block instance. The character string after the equal sign specifies the filename of the external drawing. When no character string present, however, the name of the Block will be taken as the default filename for the external drawing.
- wrkname*** An optional file name, immediately after the equal sign, specifies where to load the external drawing file to define the Block. If the extension of the drawing file is explicitly given as "DWG", it will be loaded using **DWGIN** command. Otherwise, it will be loaded as the WRK file, and an extension of "**WRK**" will be assumed if it is omitted. If this portion is not given, the default filename of the drawing file will be the same as that of the Block.

If an external drawing file, either by default or being specified explicitly, can not be found by its name, **TwinCAD** will pop up a file window for you to select an alternative one instead.

You may also specify an external drawing file reference from the slide window provided for Block name selection by selecting the **<NewBlock>** item from within it. After you enter the name of the new Block, **TwinCAD** will also pop up the file window for you to select the external drawing file to load in, which defines the Block consequently.

During the loading of an external drawing file, **TwinCAD** will solve the possible name conflicts in the ways as you issue the **LOAD** or **DWGIN** command to merge in another drawing. See also **LOAD** and **DWGIN** commands for the details.

If the Block name specified explicitly is already defined, and an external drawing file is also specified to load in, the Block will be redefined by the loading of the drawing file. **TwinCAD** will report this fact and regenerate the drawing after the Block is redefined.

After the Block is well specified, **TwinCAD** will ask you to designate the base point, and specify the scale factors and rotation angle, of the insertion, as the prompts in sequence:

```

Insert base point:
X scale factor <1>:
Y scale factor <default=X>:
Rotation angle <0>:

```

The scale factors for drawing coordinates in both X and Y axes can be different.

If the block definition being reference contains variable attribute tags, **TwinCAD** will open the attribute tag window for you to enter attribute values to each of them.

## Insert External Drawing without Building Block

If you specify the block reference in the following format:

```
*={WRKFILE }
```

then the work file will be loaded, rotated, scaled, and translated as required without building the Block definition. It functions as the **LOAD** or **DWGIN** command but with more flexibility.

So, to merge in a work file with a scale factor to a given location, you may issue this **INSERT** command and then enter "\*"=" string. **TwinCAD** will open the file window for you to specify the file to merge in. Then, follow the same procedure as that for a common insertion.

## Special Notes

If the content of the system variable **EXPERT** is 0, the **INSERT** command will ask for user confirmation if an existing block is to be re-defined by an external drawing. This confirmation is done in GUI manner, and will not affect the normal script processing.

### Procedure:

Enter the **INSERT** command to **TwinCAD** prompt:

@CMD: **INSERT** *(return)*

- **To insert an existing block: TESTBLOCK**

Block name or (?) <...>: **TESTBLOCK** *(return)*

Insert base point: *(point)*

X scale factor <1>: *(value)*

Y scale factor <default=X>: *(value)*

Rotation angle <0>: *(value)*

...

- **To insert an existing block: TESTBLOCK and explode it afterwards**

Block name or (?) <...>: **\*TESTBLOCK** *(return)*

Insert base point: *(point)*

...

- **To insert NEWBLOCK by external drawing file: DRAWFILE.WRK**

Block name or (?) <...>: **NEWBLOCK=DRAWFILE***(return)*

If the DRAWFILE.WRK does not exist, a file window will pop up, and you can select another drawing file to define the NEWBLOCK.

Insert base point: *(point)*

...

- **To redefine TESTBLOCK by external drawing file: DRAWFILE.WRK**

Block name or (?) <...>: **TESTBLOCK=DRAWFILE***(return)*

If the DRAWFILE.WRK does not exist, a file window will be pop up, and you can select another drawing file to define the NEWBLOCK.

Redefine block TESTBLOCK by file: DRAWFILE.WRK  
*(drawing regeneration)*

Insert base point: *(point)*

...

- **To insert a block by selection**

Block name or (?) <...>: *?(return)*

A slide window pops up, and you select one of the defined block.

Insert base point: *(point)*

...

- **To insert a block by selection, and explode it afterward**

Block name or (?) <...>: **\*?(return)**

A slide window pops up, and you select one of the defined block.

Insert base point: *(point)*

...

# INVGEAR

## Involute Gear Shape Generation Command (Option)

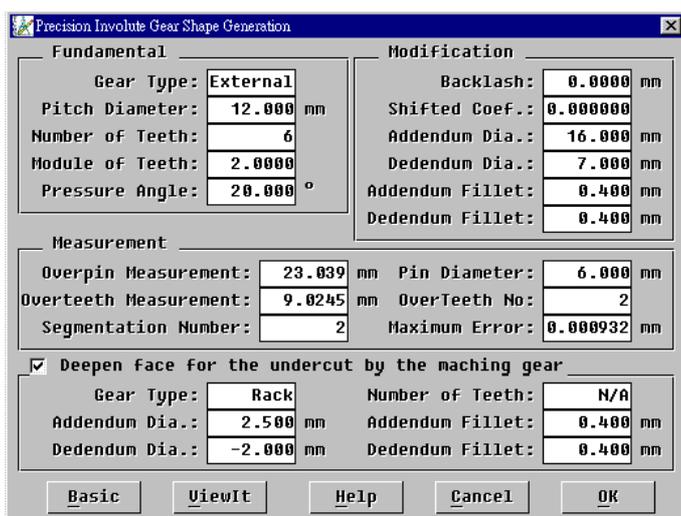
### Purpose:

The **INVGEAR** command is used to create the involute gear shape to desired accuracy and precision, specific for CAM application. This is an **Optional** command package which requires a separated authorization.

### Description:

The **INVGEAR** command lets you draw the involute gear shape of different kind of specification to your desired accuracy and precision. The generation of this gear shape may be specified to a minimum as a half of a tooth, or to a maximum as a whole gear, from any starting angle with respect to the gear center.

Before generating the gear shape, **TwinCAD** will pop up a parameter setup window, as shown in the figure below,



for you to specify the requirements of the gear shape, which can be categorized as below:

- **Fundamental Requirement** -- The basic elements that specify the shape of the gear, including *tooth type*, *pitch diameter*, *number of teeth*, *module of teeth* and *pressure angle* of the gear.
- **Modification Requirement** -- The modifications of the gear shape required other than the fundamental for specific purpose. These include the *backlash* amount, *shifted coefficient* for gearing purpose, the addendum and dedendum diameters and the fillet radii of the both with the tooth surface of the gear.
- **Measurement Requirement** -- The measurement of the resulting gear shape can be calculated or specified by the **Overpin Measurement Method**, or by the **Overtooth Measurement Method**. Setting up this requirement will force **TwinCAD** to recalculate the modification factor (shifted coefficient) needed to meet it. The precision of the gear shape generation control is also in this category.
- **Undercut Checking Requirement** -- You can setup a matching gear which is used to match with the generating gear (the one you are to produce) for further undercut and

interference checking. You can specify to deepen the generating gear face to accommodate the undercut produced by the matching gear. This is a new feature implemented after V3.1.057 in Windows version only.

The setting up of these parametric requirements requires some knowledges about the formation of an involute gear, and will be discussed briefly in later sections.

Once you have set up the proper requirement for the gear shape generation, you should pick up the screen button labeled as [ **OK** ] to continue the operation. **TwinCAD** will generate the required gear shape of one tooth and prompt:

Enter center position of gear:

asking you to indicate the center position of the gear. And then, you will be asked to specify the starting angle of the gear tooth:

Alignment/<Starting Angle of Gear(0°)>:

where the alignment of the gear tooth with respect to the starting angle, either from the addendum or the dedendum, is controlled by toggling the sub-command option "**A**". After the starting angle is specified, **TwinCAD** continues to prompt:

Span/<Number of Teeth (+=CCW, -=CW)(*Tn*)>:

asking you to specify the number of teeth to generate. The default number is the teeth number for the whole gear. Any number greater than this default number is also taken to specify the whole gear. The number of teeth to generate is not necessary integer value. It can be specified to the multiple of 0.5 unit, since the generation of the gear shape is in unit of a half tooth. If only a part of gear is to be generated, the sign of the teeth number determines the direction of the generation. Positive number generates counter-clockwise, and negative number, clockwise, respectively.

You may specify the teeth number to generate by the total spanning angle of these teeth. Enter the sub-command option "**S**", and then enter the angle value to the prompt:

Spanning angle (+=CCW, -=CW)(360°):

Again, the span angle is rounded to the nearest multiple of the half-tooth span angle.

The generated gear shape will be in the form of polyline. If a whole gear is generated, the polyline will be closed. The involute tooth shape is approximated by successive arc segments smoothly joined together, passing through the dividing points of the ideal involute curve. The tangent vector direction at these points are also maintained.

## Involute Gear Shape Parameter Setup

The Involute Gear Shape Parameters define the geometry of the gear. Changing the parameters changes the shape of the gear. Some of these parameters are correlated. Changing one of them will force recalculation of the other related ones. For example, changing the Pitch Diameter and the Number of Teeth from the *Fundamental Requirement*, will change the Module of Teeth, and also the Addendum/Dedendum Diameter and Fillet Radii from the *Modification Requirement* are recalculated for their default values. All the parameters in *Measurement Requirement* also receive recalculation, automatically.

Therefore, the proper procedure in setting up these parameters should be in the order:

1. **Fundamental Requirement** -- These are the basis of the gear shape. Changing the parameters from this requirement will definitely force the default values for the parameters in the other requirements be recalculated. So, this should be set up first.
2. **Modification Requirement** -- After setting up the fundamental requirement, you may modify the default parameter values from this requirement. Note that changing some of

these parameters will also force the parameter values from the *Measurement Requirement* be recalculated.

- 3. **Measurement Requirement** -- Usually, the measurement parameters in this requirement are taken for reference only. However, if you should change one of them, the Shifted Coefficient from the Modification Requirement will be recalculated so that the specified measurement requirement is met. The other measurement parameters will also be updated for the change of the Shifted Coefficient from the modification requirement.
- 4. **Undercut Checking Requirement** -- If you need to check whether the gear shape generated under the above requirement settings will suffer undercut with a specific matching gear, you should setup the parameters of the matching gear, in the final step.

After setting up these requirements, you may specify the *Segmentation* Number located under the Measurement Requirement such that the maximum error of the gear shape generation falls within your requirement, and thus complete the setup.

### Pitch Diameter, Number of Teeth and Module of Teeth

The relationship among these three parameters follows the equation:

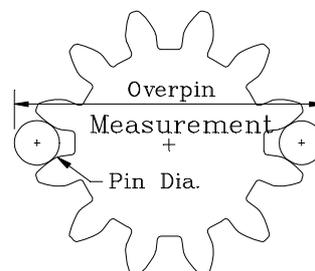
$$\text{Pitch Diameter} = \text{Number of Teeth} * \text{Module of Teeth}$$

So, setting any two of the three parameters will force the third one be recalculated by the above equation.

### Overpin Measurement and Pin Diameter

**Overpin measurement** is a method to measure the geometry of a gear. It employs two pins of the same diameters, used as auxiliary tools to help out the measurement. You have to enter the proper diameter value of the pin used for the measurement. There is no checking about the adequacy of the pin diameter against the geometry of the gear teeth.

Based on the pin diameter used and the geometry information about the gear shape, the theoretical overpin measurement value is calculated and displayed in the field of **Overpin Measurement**.

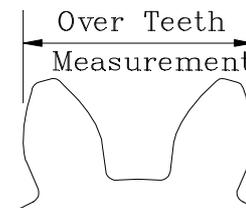


Should the specification of the gear wanted involve the required overpin measurement result, you can enter the required value into the field of *Overpin Measurement* with the specified pin diameter. **TwinCAD** will automatically adjust the shifted coefficient of the gear from the Modification Requirement, such that the resulting overpin measurement will meet the specification.

### Overteeth Measurement and Overteeth Number

**Overteeth measurement** is also a method to measure the geometry of a gear. It measure the distance between two tooth surfaces. The two tooth surfaces may belong to the same tooth (over one tooth), or each belong to different tooth (measurement over several teeth). You have to specify the number of teeth that will be measured over using the method. There is also no checking about the adequacy of this overteeth number.

Based on this overteeth number and the geometry information about the gear shape, the theoretical overteeth measurement value is calculated and displayed in the field of **Overteeth Measurement**.



Should the specification of the gear wanted involve the required overteeth measurement result, you can enter the required value into the field of Overt teeth Measurement with the specified overteeth number. **TwinCAD** will automatically adjust the shifted coefficient of the gear from the *Modification Requirement*, such that the resulting overteeth measurement will meet the specification.

## Segmentation Number and Maximum Error

The theoretical involute curve of the tooth surface has to be approximated by a successive arc segments joined together. The number of the arc segments used in this approximation is controlled by the **Segmentation Number**, which specifies the division count of the involute curve.

This division count can be zero, which means there is no division of the involute curve and the whole curve is approximated by two arcs (spline arcs). A division count of 1, will make the involute curve be divided into two portions, each is approximated by two arcs; so, there will be four arc segments used in total. A division count of 2, 6 arc segments will be used in the approximation of the 3 portions of the curve.

In short, a division count of N, there will be a total of  $2*N+2$  arc segments used in the approximation. Note that, for best error control, the division is not done evenly over the interval of the curve.

Once the segmentation number is given, **TwinCAD** will calculate the maximum error that the approximation will result in from the theoretical curve. The larger the segmentation number is, the more arc segments are used in the approximation, and the less maximum error is introduced.

Since the calculation of the maximum error may be time consuming, it is calculated only when the **Segmentation Number** or the **Maximum Error** field is activated by the cursor.

## New Features after V3.1.057

### The Generating Gear vs. the Matching Gear

The Generating Gear is the gear to generate by **INVGEAR** command. It is specified by the "Fundamental", "Modification" and "Measurement" parameters.

The Matching Gear is the pseudo gear used to match with the Generating Gear for undercut and interference checking. It is specified by the group leaded with the option checkbox:

#### Deepen face for the undercut by the matching gear

Most of the gear parameters of the matching gear must be the same as the Generating Gear so that they can match smoothly. However, you may specify the following:

1. Gear type: You can match an external gear with another external gear, a gear rack or an internal gear. However, you can only match an internal gear with an external gear.
2. Number of teeth: If you select a gear rack, this field is not applicable. However, for external or internal gear, you must specify the teeth number.
3. Other modifiable data: Same as the data in the "Modification" group for the Generating Gear, the addendum/dedendum and fillet values for the matching gear can also be modified after the default calculation. Note that if it is the gear rack in selection, the addendum/dedendum value will be the height of the gear top and bottom with respect to the center pitch line.

### Modify gear shape by the result of undercut checking

Should an undercut happen when the generating gear matches with the given matching gear, **INVGEAR** will not automatically modify the gear shape to accommodate undercut by the matching gear, unless you specify it with a check in the leading checkbox:

**[ ] Deepen face for the undercut by the matching gear**

Note that, if you do not have a particular gear to match with the generating gear, use the gear rack for the undercut checking, since it will produce the maximum undercut.

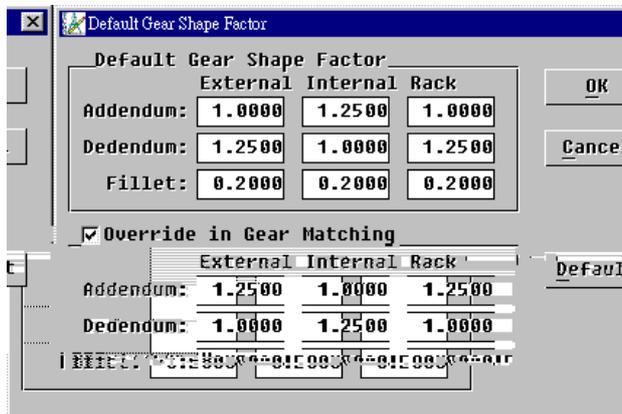
If the generating gear is an internal gear, the matching gear must be an external gear smaller than it. However, the matching may result in gearing interference other than the undercut if the pinion is relatively large. The current version will not automatically cut the gear face for such interference. In fact, this function will be disabled if the generating gear is an internal gear.

### The new [Basic] Button

The new button labeled as "**Basic**" is used to setup the default factors used in the gear shape calculation.

When you specify the gear by its fundamental data, such as the gear type, tooth number, pitch diameter and module, **INVGEAR** will automatically calculate its "Standard" addendum and dedendum diameter values, and their default fillet radii with the gear face. This calculation is based on some "Standard" factors times the module.

However, you may change this "Standard" by pressing the **[BASIC]** button and modifying these factors from the dialog window pop-up as shown below:



There are two groups of factors to setup; namely, they are:

1. Default Gear Shape Factors -- These factors are used for both the generating gear and the matching gear by default.
2. Override in Matching Gear -- These factors are used optionally for the matching gear only, to override the default gear shape factors. There is a checkbox before the group name. Check it to apply this group of factors to the matching gear.

Each group of factors has three column data entries for the external, internal and rack gear respectively, and each column has three data fields for the addendum, dedendum and fillet radius respectively. If one chooses to generate an external gear in the "Fundamental Setup", it will be the set of factors in the "External" column used in the standard calculation.

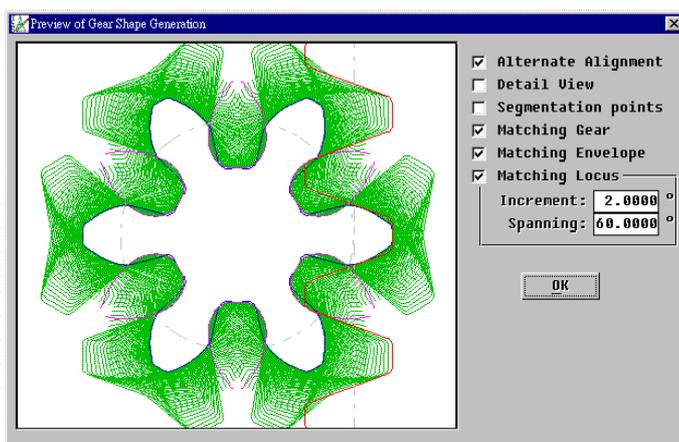
The use of the second group of factors is to override the calculation for the matching gear, which is used for the undercut checking. It is used when the matching gear is assumed to be a cutting tool which is apparently having a different measurement in its addendum and dedendum values. Of course, one easy solution is to use the standard factors interchangeably for the addendum and dedendum calculation. But, to make it more flexible to the user, a second group of factors is employed.

The **[Default]** button will reset these factors to the internal defaults.

## The new **[ViewIt]** Button

The new button labeled as "**ViewIt**" is used to provide a preview of the gear shape of the generating gear as well as the matching details with the matching gear.

Before generating the gear shape, you may press the **[ViewIt]** button to preview the gear result in advance. **INVGEAR** will pop-up a dialog containing a preview sub-window and a few option checkboxes, as an example shown below:



These options are described in details from top to bottom as below:

**Alternate Alignment** If this option is checked, the gear view will be generated such that the center of the addendum top is aligned with the X-axis, in terms of an external gear. Otherwise, it is the center of the dedendum bottom aligned with the X-axis.

**Detail View** If this option is checked, the gear view will be generated such that the details of the first gear counted from the +X axis can be clearly observed. It will be at the center of the view and zoomed to an appropriate extent to provide a clear observation. Otherwise, it is the whole gear at the center and zoomed for the whole view.

The generating gear is plotted in blue.

**Segmentation points** If this option is checked, the segmentation points along the gear face will be marked with a tiny arrow pointer. Otherwise, they will not be shown in the view.

**Matching Gear** If this option is checked, the shape of the matching gear will also be plotted in the view. The matching contact will be at the first gear counted from the +X axis of the Generating Gear.

The matching gear will be plotted in red.

**Matching Envelope** If this option is checked, the maximum matching envelope generated by the matching gear with respect to the generating gear will be plotted. This envelope curve is calculated internally by formula deducted from the gear matching motion and is used for the required undercut checking and shape calculation.

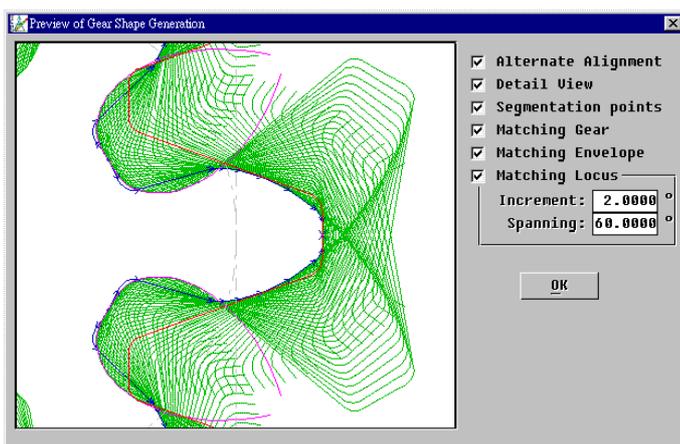
The matching envelope will be plotted in cyan.

**Matching Locus** If this option is checked, the locus of the gear matching motion simulation will be plotted out for visual checking. You can specify the increment angle as well as the spanning angle of the motion in the data fields following this option checkbox.

Note that the matching envelope is plotted via the formula deducted from the gear motion, not by the simulation result. While the matching locus shows the result by direct simulation, you can always check between them and be sure that the result is correct.

The matching locus will be plotted in green.

An example of the detail matching view is given below:



### Limitations

Currently, there are certain limitations not resolved yet, but may be resolved in the future:

1. Interference of Gearing between an internal generating gear and an external matching gear is not removed by deepening the gear face. Usually, this is avoided by using stub gear teeth.
2. Valid ranges of each parameter regarding gear shape dimension are not checked. Ridiculous result may happen if you should have supplied wrong data.

### Procedure:

- **An unauthorized use of this command**  
 @CMD: **INVGEAR** (return)  
 Command not authorized to use, Ignored.  
 If you need this involute gear generation package, please contact to your local dealer for details about the price of license of agreement.
- **An authorized use of this command**  
 @CMD: **INVGEAR** (return)

*(Set up parameter window operation)*

Enter center position of gear:

Alignment/<Starting Angle of Gear(0°)>:

Span/<Number of Teeth (+=CCW, -=CW)( $T_n$ )>:

**Example:**

**'ISOPLANE**

## Isometric Projection Axes Setup Command

---

**Purpose:**

The **ISOPLANE** command is used to set up the projected axes directions for an Isometric Projection.

**Description:**

The **ISOPLANE** command lets you set up the projected axes directions for an Isometric Projection and select the active Isometric plane.

When the **ISOPLANE** command is entered, **TwinCAD** will check whether the current axes setup is one of the Isometric Projection Axes setup or not. If it is not, **ISOPLANE** will initialize it with the Isometric/Left index 0 (L-0) plane axes setup. The PUCS mode will be enabled automatically and you will see the Axonometric Axis Tripod shown in the screen (at current PUCS origin).

And then, **TwinCAD** will prompt

```
Isoplane -- Left/Top/Right/<plane:L-0/base:(0.,0.)>:
```

for you to change the Isometric PUCS-Plane and its base point (the PUCS origin). Once a change is made, you may directly observe it from the screen (by the change of axis tripod and the crosshair orientation), and **TwinCAD** prompts again. You have to type space bar or <Ctrl/C> to exit the command.

You may answer to the prompt with the following input options:

- <point> Point designation, specifying the new origin of the PUCS-plane. The cursor snap base point will be reset to (0,0), and the PUCS mode will be enabled if it is currently disabled. You may disable the PUCS mode by the <Ctrl/U> function.
- <n> Integer number from 0 to 3, specifying the index number of the current Isometric PUCS-Plane setup.
- L Left, specifying the Isometric/Left PUCS-plane.
- T Top, specifying the Isometric/Top PUCS-plane.
- R Right, specifying the Isometric/Right PUCS-plane.

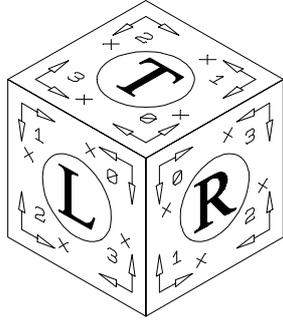
See also **AXOPLANE** command reference.

### Isometric PUCS-Planes

The cube under Isometric Projection, as seen from the figure below, shows three projection planes. These planes are called the Isometric Projection Planes. Each of the planes is identified as the Left-plane, the Top-plane and the Right-plane, as the letters shown on the plane surfaces, respectively. Observing that each plane surface on the cube has four corners, each is identified by an index number starting from 0 at the center and counting counter-clockwise around.

To set up the Isometric Projection Axes for the PUCS-plane operation, it is apparently that we can have four choices (assuming only right-handed PUCSs are used) to set up the axes for each of the Isometric Projection Plane. And, in total, we can have 12 different axes setups (actually is 24, if left-hand system is included).

The 12 Isometric UCS-planes are thus identified by **L-0, L-1, L-2, L-3, R-0, R-1, R-2, R-3, T-0, T-1, T-2** and **T-3**. You may use the **<Ctrl/I>** function to toggle this Isometric UCS-Plane. The toggling sequences will follow the order: **L-0, R-0, T-0, L-1, R-1, T-1, L-2, R-2, T-2, L-3, R-3, T-3**.



### Automatic Foreshortening Factors Setup

If the auto-scaling function is enabled, the three foreshortening factors will be set to 1, and the value of the system variable **MVSCALE** will be set to **sqrt(1.5)** automatically. The system variable **AXOSCALE** will also be set to the same value as the **MVSCALE**.

Otherwise, if the auto-scaling function is disabled (**AXOSCALE** is zero), the three foreshortening factors will be set to  $1/\sqrt{1.5}$ , and the value of **MVSCALE** will be set to zero.

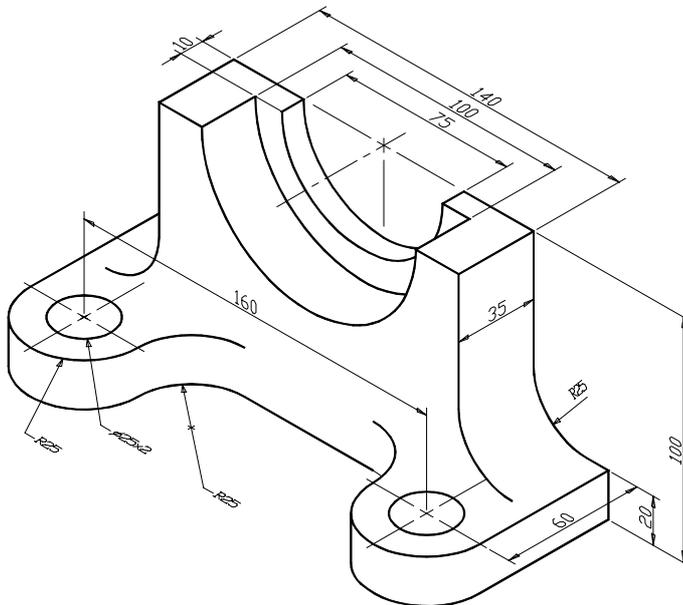
### Procedure:

@CMD: **ISOPLANE** (return)

Isoplane -- Left/Top/Right/<plane:L-0/base:(0.,0.)>:

...

### Example:





# 'LAYER

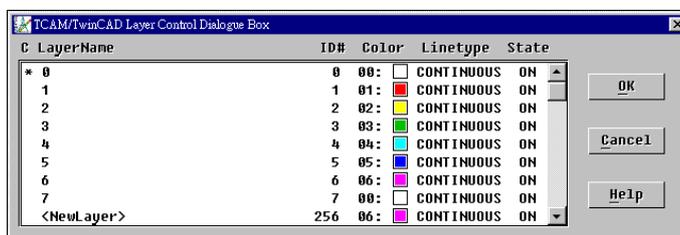
## Drawing Layer Control Command

### Purpose:

The **LAYER** command is used to access the drawing layer control via a dialogue window operation manner.

### Description:

The **LAYER** command lets you control the drawing layers. When you issue **LAYER** command, **TwinCAD** will pop up a dialogue window for you to create layers, view all of them, modify their default properties and select the active layer from the window. These are described in the following paragraphs. An example of the layer dialogue window is given below:



### Create New Layer

To create a new layer from the window, pick up the **<NewLayer>** item from the column of layer name display, and then, enter the name of the new layer to create. If you type characters in lower cases, they will be converted to upper cases. At most 10 characters can be entered to the field.

A newly created layer will be given an initial default setting of color, linetype and status, assigned with an internal ID number, and become the current active layer. You can subsequently modify these setups.

Note that the layer **0** is always created automatically when a drawing is initially started. The maximum number of layers are limited to 256.

### Select Active Layer

To switch current active layer to a specific one, just pick up the layer's name you want to switch to. An '\*' (asterisk sign) will be marked ahead of the row of the current active layer. All subsequent drawn entities will be on this active layer.

### Modify Layer Color

To modify a layer's default color, pick up the color field on the row of the layer you want to change. **TwinCAD** will pop up a slide window of color selection. You simply select the color you desire for the layer from it.

### Modify Layer Linetype

To modify a layer's default linetype, pick up the linetype field on the row of the layer you want to change. **TwinCAD** will pop up a slide window of loaded linetype selection. You simply

select the linetype you desire for the layer from within it. If the linetype you desire is not in the linetype selection window, use **LINETYPE** command to create or load it. See also **LINETYPE** command reference.

### Alter Layer Status

To alter the layer's status, pick up the status field of the layer you want to change, and **TwinCAD** will pop up a status selection box for you to select the desired status for it.

Currently supported status for a layer are **ON** and **OFF**. When a layer is in **OFF** state, all the entities belong to the layer will not be drawn.

If you have changed any layer's status, after you exit the window, **TwinCAD** will regenerate the screen to reflect the change.

To exit the dialogue window, press <**ESC**> key from keyboard, or pick the upper left [-] button from the window, or depress the right button of your mouse.

### Procedure:

To access the drawing layer control, type **LAYER** to the command prompt:

@CMD: **LAYER** (return)

### Example:

**LDIM**

## General Linear Dimensioning Command

---

**Purpose:**

The **LDIM** command is used to create linear dimensions.

**Description:**

The **LDIM** command lets you create linear dimensions. It is the formal entry command for the linear dimension commands: **HDIM**, **VDIM** and **ALDIM**. See the description of these commands for details.

**Procedure:**

Enter the **LDIM** command to **TwinCAD** command prompt:

@CMD: **LDIM** *(return)*

and **TwinCAD** will prompt,

Horizontal/Vertical/Aligned/(H)<First dimension point>:

assuming the last linear dimensioning is horizontal. If the last linear dimension mode is in vertical dimensioning, then the prompt will be:

Horizontal/Vertical/Aligned/(V)<First dimension point>:

showing that you are in vertical linear dimensioning.

**Example:**

# LEADER

## Create a Leader Line/Spline Command

---

### Purpose:

The **LEADER** command is used to create a leader line or leader spline.

### Description:

The **LEADER** command lets you create a leader line. You create such a leader by designating the start point of the leader first, and then, drag the leader line segment out by designating more points.

At most 4 line segments (5 node points) can be created after the start point. However, you may use **LINE** command to generate more line segments. After the leader line is created, You will have to add **TEXT** by yourself.

The leader stops automatically when the fourth line is drawn. Before the leader stops, you may enter the sub-command option "**U**" to cancel or displace the last leader segment.

You may use **SETDIM** command to set or change the arrow pointers on both ends of the leader lines. Initially, only the starting point has the default arrow pointer set by **DIMATYPE** variable. You may also use **QCHANGE** command to modify the position of its node points directly. Note that leader is also a dimension entity in **TwinCAD**.

### Creating Leader Spline

You may create a leader spline by setting bit 8 of the dimension variable **DIMLOPT** to be 1 (ON), and then issue the **LEADER** command. After you have created a leader line, **TwinCAD** will transform it automatically into leader spline, according to the following rules:

- The node points of the leader line will be taken as the control points of the spline applying the continuous Quadratic Bezier Curves. The resulting curve will be the same as that produced by the **SPLINE** command over the equivalent polyline.
- If two neighboring control points should coincide, straight line segments will be resulted on both sides as a natural consequence of the conversion. The **LEADER** command allows you to duplicate the node points during the creation of a leader line.
- If two neighboring line segments should be collinear to each other, then a straight line segment will also be produced as a natural result of the conversion.

To set the bit 8 of the **DIMLOPT** from the command line, input the expression below to the command prompt:

```
@DIMLOPT|=0x100
```

and to reset it, input the expression

```
@DIMLOPT&=-0x100
```

Once a Leader Spline is created, you may use **QCHANGE** command to modify the position of its node points and thus dynamically change the shape of the spline curve, if necessary.

### Special Notes

You may now use **SLEADER** to create spline leader in **TwinCAD** for Windows. See also **SLEADER**.

**Procedure:**

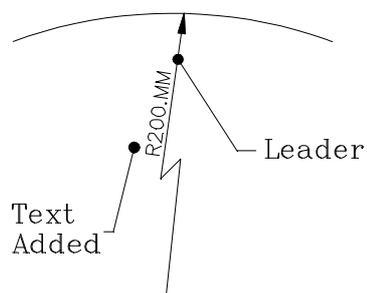
Type **LEADER** to the command prompt, and follow the procedure as below:

@CMD: **LEADER** (*return*)

Indicate pointer location: (*point*)

Lead line to: (*point*)

Undo/<Lead line to>: (*point*)

**Example:**

**LIMIT**

## Set Drawing Limit Command

---

**Purpose:**

The **LIMIT** command is used to set the intended drawing limits.

**Description:**

The **LIMIT** command lets you set up the drawing limits. The drawing limits define the coordinate ranges where the drawing entities can be created. This is an **AutoCAD**-compatible command. However, it is effective only for **GRID** generation and **ZOOM** by limit command; there is no limit checking over the drawing entities in creation.

**Procedure:**

Type **LIMIT** to the command prompt, and follow the procedure as below:

@CMD: **LIMIT** (*return*)

First corner: (*point*)

Second corner: (*point*)

**Example:**

# LINE

## Line Entity Creation Command

---

### Purpose:

The **LINE** command is used to draw lines in 2D and/or 3D space.

### Description:

The **LINE** command lets you create Line entities in several ways.

### Two Points Make A Line

Basically, a line is defined by two given end points. Any method that specifies a point can be used to specify the end points, such as

- An absolute point position, such as the **END**point of an arc segment, **MID**dle point of a line segment, **CEN**ter point of a circle, direct coordinates input, and so on.
- A relative point position (relative to last point)
- A **TAN**gent point to a specific object (Circle, Arc, Ellipse)
- A **PER**pendicular requirement (normal) to a specific object (Line, Circle, Arc, Ellipse)

For example, you may specify the start point of the line to be the tangent point on a circle, and the end point, on another line such that the two lines perpendicular to each other. To do so, You just need to pick up the first circle by the **TAN** snap function, and the second line by the **PER** snap function, or in reverse order.

### Dragging of Solution Line

**TwinCAD** provides very powerful dragging capabilities, so that you can see what you are doing. For example, if you specify the start point as **PER** to a line (using **PER** snap function and picking at a line), which means the resulting line should be perpendicular to the specified line and the start point must be also on it, then you can see that the dragging line, from the cursor position to a point on that specified line, is always perpendicular to it with one end point sliding along it as the cursor position changes. You can always drag a line of any kind of specifications that are supported in such a manner.

### Coordinate Display of the Next Point

During the line-drawing process, you will be guided to enter the next points by current cursor coordinates which appear in the Status Line area at the top of the screen. The initial setting for the coordinate display will be in distance/angle format (i.e., polar coordinate of the next point relative to the previous point).

However, you may type **<Ctrl/E>** during the operation to make switch between polar and Cartesian coordinates. This coordinate information, when combined with the dragging functionality, provides a convenient way for you to enter the next points in the line construction process.

## Additional Constraint after First Point -- Fixing A Line Direction

A line with a start point specification can be fixed in a direction by the use of sub-command options, "**DI**" and "**D**". They are used to specify the absolute angle direction and the relative angle direction of the line to be drawn, respectively.

The absolute angle is an angle relative to the X-axis in counter-clockwise direction, while the relative angle is an angle relative to the tangent direction of a given object at the start point (which must also be on that object). This tangent direction being used as a relative reference is called the reference angle.

The determination of the reference angle depends on how the start point is specified. If the start point is determined from a given object (e.g., by **MID** of, **END** of, **QUA** of, **NEAr** to, etc.), then the direction angle of the object at that point will be taken as the reference angle. This also applies to the line that is continued from the last line/arc created.

If there is no object selected with the start point specification, then the reference angle will be zero. In such case, the function of "**D**" will be the same as "**DI**".

However, if the start point is specified by the use of **TAN/PER** snap function to a given object, since the position of the start point is not yet resolved and the reference angle may not be calculated immediately after it is specified, the sub-command options "**DI**" and "**D**" will be used to modify the meaning of the **TAN/PER** in the start point specification, depending on the type of object specified:

- **PER to line - *Start point on a line at a given angle*** (absolute or relative to the line). Initially, the dragging line is perpendicular to the specified line, with the start point sliding along it. This means, the resulting line is already in fixed direction of 90° relative to the specified line. The "**DI**" and "**D**" options can be used to change this direction while retaining the start point sliding along the specified line.
- **PER to circle - *Normal to a circle at given direction***. Initially, the dragging line is perpendicular to the specified circle, with the start point sliding along it; however, the direction of the line is always changing as the cursor moves. The "**DI**" and "**D**" options will fix the line in the given direction, while it is perpendicular to the circle. The start point is thus determined by the nearest intersection of the line with the circle to the picking point.
- **TAN to circle - *Tangent to a circle at given direction***. Same as above, except the line is tangent to the circle. The start point is determined such that the line is fixed in direction and tangent to the circle.
- **PER to ellipse** - same as **PER to circle**, except the object is an ellipse.
- **TAN to ellipse** - same as **TAN to circle**, except the object is an ellipse.

Note: If the PUCS-plane is active, the angle input for the "**DI**" and "**D**" options will be the angle measured on the projected plane from the virtual space.

After version V3.1.056, an addition sub-command option "**Through**" is provided to support the specification of a line through a given point, after the first point is given. The prompt sequences are:

```
Direction/Deflect/Free/Through/<To point/length>: T (return)
Through point: (point)
Direction/Deflect/Free/Through/<To point/length>:
```

The line creation will be fixed in the direction such that the given point will be passed through geometrically.

Note that once a through point is specified, the direction will be the base reference angle for the subsequent Deflection angle. The sub-command "**Free**" is also effective to release this special constrain in line creation.

Note also that if a null reply is given for the through point, the through point constrain will also be released.

### Special Usage of PER

The **PER** snap function has another special usage in drawing the line. If the direction of a line has been fixed either by specifying the angle direction with the "**D**" or "**DI**" options, or by the nature of default, you may define its end point by specifying it **PER** to another object, such that the line will be extended in the fixed direction to **TOUCH** right on the selected object.

Use the "**DI**" or "**D**" options in combination with the **PER/TAN** snap functions, it is quite easy to draw a line tangent to a circle at a given angle with one end point on another line.

### Specifying Length of a Line Segment

When a line has been fixed in one direction and the start point is known already, you may specify the end point by the length of the line. You specify the length by replying to the prompt a single value, instead of a point. If you designate a point, in this case, the point projected on the line in that given direction is taken.

In fact, if the start point is already known in a fixed position, you may specify the length first before fixing the line direction. You may drag the line segment of fixed length rotating around the starting point until you complete the end point specification.

### Special End Point Specification

For all start point specifications except the one **PER** to a line, relative coordinates (**@len<ang** or **@dx,dy**) can be given to the end point specification. If the start point is **TAN/PER** to a circle, the direction of the line specified by the relative coordinates implicitly determines the start point on the circle.

### Continue to Draw Lines

After a line is drawn, **TwinCAD** will continue the line drawing command automatically and take its end point as the start point of the next line to create. The direction of the previous line is taken as the reference angle (see previous paragraphs). After several lines are drawn, you may use the "**C**" option to close the drawing lines to the first point at the very beginning of this command. This help you create polygons.

Also, if you reply a null return to the start point specification, **TwinCAD** will assume that you are continuing the last entity creation, and uses the last point as the default start point.

To exit the command at anytime, reply to the prompt by a null return or type **<Ctrl/C>**.

### Undo the Last Line

In continuing to drawing lines, you can undo the last-drawn line by the sub-command option "**U**". When you undo a line, the Last Point (used as a default start point) is also restored back to the start point of the just erased line segment. However, if there is no line segment to undo, then the start point of the very first line is undone. **TwinCAD** will ask you to re-designate the first point of a line.

Note that when the very first line is undone, the Last Point will not be restored back to the point before the first line is drawn, but rather, the end point of the first line segment is taken as the Last Point. This feature can be used to locate a point on a object that can not be directly designated. For example, to determine a point on a given line at a given distance from one end point of the line, you may issue the **LINE** command, pick up the ENDpoint of the line, use "**D**" to set the direction, enter the distance required, and then use "**U**" to undo the line just drawn. Now, the Last Point is the point you want. You don't need to bother with the **ERASE** command.

## Elevation And 3D-lines

If you designate a point by picking on an object, the elevation of the object at the snap point will be the Z component of the point, as if you explicitly type it from the keyboard. Otherwise, the Z component will be default to the current cursor elevation, which is initially set to the current Entity Elevation.

If the two end points of a line lie at the same elevation, it will be a 2D-line on the X-Y plane at the elevation. Otherwise, it is a 3D-line.

You can't produce a line by **PER** to two given lines on the same plane. However, you absolutely can produce such line from two lines that do not lie on the same spatial plane. The line produced in such way will be the shortest distance between the two given lines.

## Sub-Command Options

A summary on sub-command options is given below:

- **DI** - Direction, request to fix the line at given direction absolutely.
- **D** - Deflection, request to fix the line at given direction relative to the reference direction.
- **F** - Free, request to free the line's constraint in fixed direction, so that the effect of **D** or **DI** is canceled.
- **U** - Undo, request to undo the last line segment.
- **C** - Close, request to conclude the **LINE** command by drawing a line to the first point where the **LINE** command is started.

## Special Notes

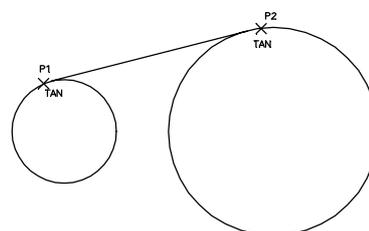
All possible solutions provided by **LINE** command in association with the ellipses are completed (as with the circles).

The only case that the solution may fail is to find a perpendicular point near to the minor axis end points of an ellipse. This is because the tangent value of the normal vector around this point (for a normalized ellipse) is very large (and is infinite at this point). The error of calculation will be magnified to an extent that can not pass the test for correctness. However, this should cause no inconvenience, since the ability to find the perpendicular point is already superfluous.

## Procedure:

The following are example procedures:

- **To draw a line tangent to two circles**  
 @CMD: **LINE** (*return*)  
 @CMD: LINE From Point: **TAN** to (*pick circle 1*)



Deflect/Free/<To point/length>: **TAN** to (pick circle 2)

- **To draw a line tangent to a circle, 45°, 20 unit in length**

@CMD: **LINE** (return)

@CMD: LINE From Point: **TAN** to (pick circle1)

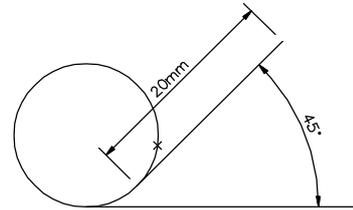
Deflect/Free/<To point/length>: **@20<45** (return)

or

Deflect/Free/<To point/length>: **D1** (return)

Direction angle from start point: **45** (return)

Deflect/Free/<To point/length>: **20** (return)



- **To draw a line intersecting a line at 45°, tangent to a circle**

@CMD: **LINE** (return)

@CMD: LINE From Point: **PER** to (pick line)

Deflect/Free/<To point/length>: **D** (return)

Direction angle from start point: **45** (return)

Deflect/Free/<To point/length>: **TAN** to (pick circle)

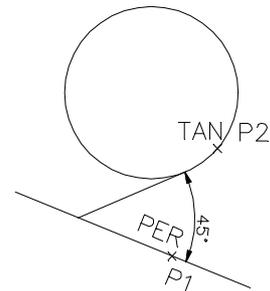
or

@CMD: LINE From Point: **TAN** to (pick circle)

Deflect/Free/<To point/length>: **D** (return)

Direction angle from start point: **45** (return)

Deflect/Free/<To point/length>: **PER** to (pick line)



- **To draw a polygon**

@CMD: LINE From Point: (point)

Deflect/Free/<To point/length>: (point)

...

Deflect/Free/<To point/length>: (point)

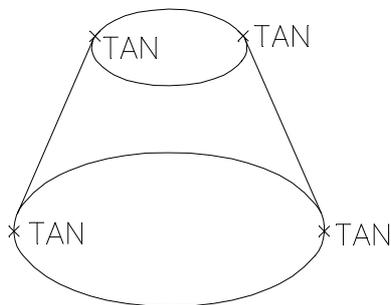
Deflect/Free/<To point/length>: **C** (return)

- **To draw a shortest line between two objects (lines, arcs, circles)**

@CMD: LINE From Point: **PER** to (pick object1)

Deflect/Free/<To point/length>: **PER** to (pick object2)

**Example:**



Example of Lines Tangent to two entities

# LINETYPE

## Load/Create/Modify Linetype Command

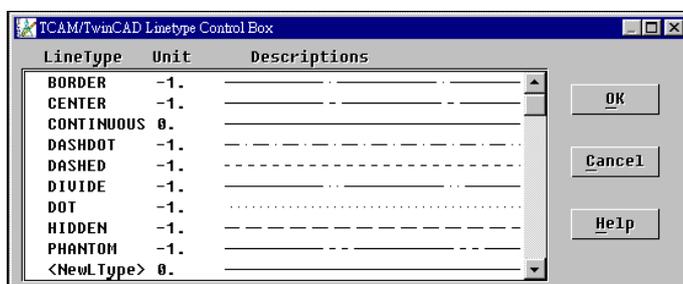
---

### Purpose:

The **LINETYPE** command is used to load or create new linetypes for subsequent use, or to modify the existing linetypes.

### Description:

The **LINETYPE** command lets you load predefined linetypes from a disk file or create new linetypes manually, or modify the existing linetypes. When you enter the **LINETYPE** command, **TwinCAD** will pop up a linetype control window, where you can load/create new linetype or modify the existing ones. An example of the linetype control window is given below:



### Create New Linetype

To create new linetype, pick up the **<NewLType>** item from the linetype control window. **TwinCAD** will open the field for you to enter the name of the linetype to create. There are at most 10 characters can be entered. After a proper name is entered, a new linetype is created with an initial default linetype definition (a solid line). You may then modify this default linetype definition to the one you desired.

### Load a New Linetype

To load a new linetype, follow the way you create a new linetype, except that you should enter nothing to the field of the name of the linetype. A null string returned from the field of the linetype name tells **TwinCAD** to open a linetype definition file, of which the name is specified in the system initial file and is default to "**TCAD.LIN**" if not specified. **TwinCAD** will read the file in and pop up a slide window containing all the linetype definitions from the file for you to pick up the one to load into the drawing.

It is recommended that you should keep useful linetype definition to the linetype file, and load it when you need it, instead of create it manually each time. Note that if you should load an existing linetype from the file, the existing linetype will be redefined. There will be no warning of this.

### Modify an Existing Linetype

To modify an existing linetype, simply pick up the Unit item or the description item of the linetype, and you will be able to modify the basic unit or the definition of it, respectively. The basic unit of a linetype is a numerical value expressed in drawing unit if it is positive and in output device unit if it is negative.



So, **TwinCAD** accepts a negative value in the basic unit, of which the absolute value is taken to specify the dashed pattern unit in millimeter (MM) to the output device.

### The Generation of Dashed Line Pattern

As the generation of a dashed line pattern involves many calculations, it is a comparative slow process in the generation of the drawing, especially when it is generated to the graphic screen for real-time performance, like the execution of a **REDRAW** or a **REGEN** command.

To improve the performance on man/machine interface, **TwinCAD** applies a fast method to generate the dashed line pattern at the sacrifice of losing accuracy and precision in the display of the line at its end points, which should receive special considerations for better appearance. However, these details will be taken good care of when they are plotted out from the plotter.

The system variable "**FASTLTYPE**" is used to control the linetype generation to the graphic screen. See **SYSVAR** command for more details.

### The Linetype Definition File

**TwinCAD** supplies an example linetype definition file: "**TCAD.LIN**", shipped with the package. You may add new linetype definition to the file or modify the content of it.

The content of this example linetype file is as below:

```
"DASHED",3.,2,1
"HIDDEN",2.,2,1
"CENTER",3.,5,1,1,1
"PHANTOM",3.,5,1,1,1,1,1
"DOT",2.,0,1
"DASHDOT",3.,2,1,0,1
"BORDER",3.,2,1,2,1,0,1
"DIVIDE",3.,2,1,0,1,0,1
```

The first quoted string is the name of the linetype. The floating point value after the string delimited by a comma is the basic unit. The rests are these integer values that define the dashed pattern.

#### Procedure:

```
@CMD: LINETYPE (return)
(Operate on linetype control window)
```

#### Example:

**LIST**

## List Out Selected Objects Command

---

**Purpose:**

The **LIST** command is used to list out the geometries and properties of selected objects from the drawing.

**Description:**

The **LIST** command lets you inspect the geometry data and properties of selected objects from the drawing.

You will be asked to select objects of interest through the object selection operation as described in **SELECT** command section. After that, all selected objects will be listed out in the order taken from the drawing database.

Although the listing may be quite length, it will be directed to the History Recording File, and you can review it from the History Viewer invoked by typing <Ctrl/Z>. Be sure to turn ON the History Recording function by typing <Ctrl/S> or **RECORD** command, before issuing **LIST** command. See also **RECORD** command.

**Procedure:**

To inspect objects, follow the procedure below:

```
@CMD: LIST (return)
Select Objects (+): (do so)
(lengthy listing generated)
@CMD: ^Z (to view the listing)
```

**Example:**

## Load/Merge In A Drawing File Command

---

### Purpose:

The **LOAD** command is used to load in a drawing file from disk.

### Description:

The **LOAD** command lets you load a drawing file in **WRK** format from disk and merge it into the current drawing. It will pop up a file window for you to select the drawing file to load in. If this is the first work file to load (include the one specified by the command line), it will become the current drawing and all variables loaded from the file will override those from the current one (possible prototype drawing); otherwise, the file is merged into current drawing and the loading of variables follows the rules described in latter sub-sections.

### Variables Conflict

During the loading of a drawing file, new variables will be created and assume the value from it, while old variables will retain their values and will not be updated. This fact is especially apparent for the layer creation upon the loading of a drawing. New layer will be created as required, but the layer that already exists, will not be altered. So, a drawing loaded by this command may looks different in thei.96 110. 12 0 0 1 thaokbmaw aaycerla(a)-12. 0 1 thula6.9(ent d(er)-6.8)-

- **R** - Replace, telling **TwinCAD** to replace the current Block definition in the drawing with the new one loaded from the disk file. All block instance will be regenerated from the new one.
- **S** - Skip, telling **TwinCAD** to keep the old Block definition in the drawing and ignore the new one from the disk file. All block instances merged from the disk file will be regenerated from the current one in the drawing.
- **C** - Change name, default, telling **TwinCAD** to load the new one into drawing and change its name later (after all are loaded). You will be asked later to specify at most two characters to prefix these newly loaded Block name in conflict.

### Encrypted Work File

If the work file to load in was encrypted, **TwinCAD** will report the fact and prompt

```
Work file xxxxxxxx was encrypted.
```

```
Enter password to decrypt:
```

asking for the decryption string. The decryption string has to be the same as that was used to encrypt the file. If the decryption string is not correct, **TwinCAD** will not be able to read the file, and the loading operation will be aborted.

### Special Notes

The **LOAD** command will not clear the drawing database before loading in the work file. To discard the current drawing and load in a new file, use the **NEW** command. To insert (merge) a work file into the drawing at a specific position with a scale factor and/or rotation angle, use the **INSERT** command. See also **NEW** and **INSERT** command reference.

### Procedure:

To load a drawing, type **LOAD** to the command prompt:

```
@CMD: LOAD (return)
```

```
File name <>: (Select on file window)
```

```
...
```

### Example:

# 'LOADOPT

## Drawing Loading Option Setup Command (TCL)

---

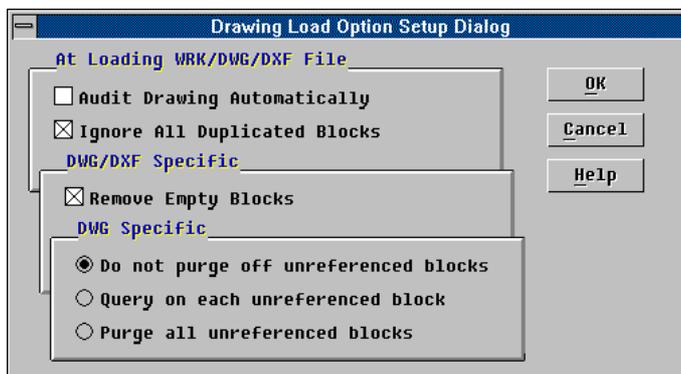
### Purpose:

The **LOADOPT** command is used to setup drawing loading related options via a GUI dialogue operation.

### Description:

The **LOADOPT** command lets you access the drawing loading related options via a GUI dialogue window operation. These options are controlled by specific system variables. With **LOADOPT**, you can easily access these options without knowing how to setup these related system variables.

**LOADOPT** will pop up the dialogue window as shown below:



The following describe these screen items in details.

### At Loading WRK/DWG/DXF File

You may specify to audit the drawing automatically at loading the WRK/DWG/DXF files by checking on the first item in this group. See **AUDIT** command for details about the drawing auditing.

You may also specify to ignore all duplicated blocks at loading the WRK/DWG/DXF files by checking on the second item in this group. Duplicated blocks are blocks that have been defined in current drawing. The **LOAD**, **DWGIN** and **DXFIN** commands have implemented rules to resolve such name conflict; however, you may override the rules to ignore all duplicated blocks by enabling this option.

### DWG/DXF Specific

You may specify to remove empty blocks from the drawing at loading DWG/DXF files by checking on the only item in this group. Empty blocks are blocks containing no geometry entities. Some **AutoCAD** add-on applications may create special blocks to store additional informations using X-data. Such blocks are meaningless in **TwinCAD**.

## DWG Specific

You may choose to purge off unreferenced blocks automatically at loading R10/R12 DWG file (this options is not yet implemented for R13/R14 at the time this document is prepared). In this special group, you may select from the radio buttons one of the following options:

- Do not purge off unreferenced blocks -- The unreferenced blocks will be loaded into the current drawing.
- Query on each unreferenced blocks -- For each unreferenced block, **TwinCAD** will query the operator whether to load it in or not, during the loading of the DWG files.
- Purge all unreferenced blocks -- All unreferenced blocks will be skipped during the loading of the DWG files.

## Other Buttons

After completing your option setup, you have to press the **[OK]** button to confirm the change and terminate **LOADOPT**. Or, you may press the **[Cancel]** to quit **LOADOPT** operation. You may also press <ESC> key or the right mouse button to quit the operation.

## Special Notes:

The **LOADOPT** command is an external command provided by the TCL program file "LOADOPT.TCL" or "LOADOPT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **LOADOPT** command, you may solve the problem by copying the "LOADOPT.TCL" to the COMMANDS sub-directory.

## Procedure:

Enter the **LOADOPT** command to the system command prompt:

```
@CMD: LOADOPT  
...[Dialogue Operation]...
```

# LRDIM

## Create Large Radius Dimension Command (TCL)

### Purpose:

The **LRDIM** command is used to create a large radius dimension.

### Description:

The **LRDIM** command lets you dimension an arc of large radius by creating a proper leader line with dimension text. You pick up an arc to dimension, designate a leader start point, and then **LRDIM** generates the required radius dimension for you.

The **LRDIM** command will prompt in the following sequence:

LargeRadiusDIM -- Select arc segment to dimension:

LargeRadiusDIM -- Indicate the starting point:

If the designated start point is outside of the arc, or it is too far from the arc (the direct distance from start point to the dimension point on the arc is larger than the arc radius), **LRDIM** will print the message:

\*\* Invalid point. You should use RDIM...

and issue the **RDIM** command for you to continue dimensioning the arc.

If the designated point is okay for a large radius dimensioning, **LRDIM** will create the dimension using leader line with a text annotation. This text annotation, i.e. the dimension text, is generated automatically in the drawing unit to reflect the true radius value of the arc. The generation of this default dimension text is governed by the dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about dimensioning.

The created radius dimensioning follows the ANSI standard.

### Special Notes:

The **LRDIM** command is an external command provided by the TCL program file "LRDIM.TCL" or "LRDIM.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **LRDIM** command, you may solve the problem by copying the "LRDIM.TCL" to the COMMANDS sub-directory.

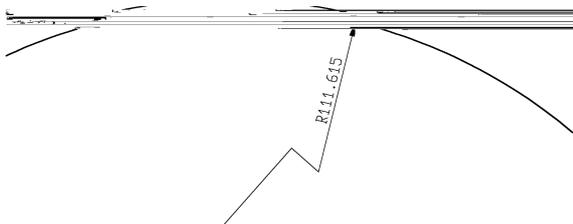
### Procedure:

@CMD: **LRDIM** (return)

LargeRadiusDIM -- Select arc segment to dimension: (do so)

LargeRadiusDIM -- Indicate the starting point: (do so)

### Example:



**'LTMASK**

## Setup Linetype Mask for Object Selection Command (TCL)

---

**Purpose:**

The **LTMASK** command is used to setup the color mask for the subsequent object selection.

**Description:**

The **LTMASK** command lets you setup the linetype mask for the subsequent object selection via a GUI dialog manner. It will open a dialog window containing 16 check-boxes for the first 16 linetypes defined in the drawing. You may place a check-mark on a linetype to specify that the objects with the corresponding linetype will be allowed to select during the object selection operation when the selection masking is enabled. See **SELECT** command for details about the object selection operation.

**Special Notes**

The linetype mask is controlled by the system variable **LTYPEMASK**, which is an integer bit-flag variable with volatile attribute and will be reset to 0 each time **TwinCAD** is initialized. Each bit of the integer from bit 0 to bit 15, corresponds to a linetype ID from 0 to 15. If a bit is set to 1, the entity with the corresponding linetype will be rejected by the object selection operation, provided that the selection mask is enabled. Linetype of **[ByLayer]** and **[ByBlock]** are resolved to the actual linetype used for the entity before the masking.

Note that the object with a linetype ID after 15 can not be masked off during the object selection operation.

**Procedure:**

```
@CMD: COPY (return)
Select Objects (+): 'LTMASK
...[Dialog Operation to setup linetype mask]...
Select Objects (+):
...
```



# MEASURE

## Measure Object Command

---

### Purpose:

The **MEASURE** command is used to measure an object with prescribed unit length.

### Description:

The **MEASURE** command lets you measure a selected object with marks placed along it at intervals of a given unit length. You will be asked to pick up the object first and then to specify the unit length of measuring.

There are several kinds of marks that you can use:

- **Point** - This is the default mark. A point entity is inserted along the object at each dividing point. You can set the system variables **PDSIZE** and **PDMODE** to view the points. See also **SYSVAR** or **POINT** command.
- **Line** - The Line marks are placed normal to the selected object. You may specify the size (length) of the line, and determine how you want to place the line marks relative to the object. The line marks can be positioned on either side of the object, or be bisected by the object in the middle. After pre-setting the line marks, you will be asked to enter the segment length of the measuring unit. The default case for placing the line marks is Middle.
- **Circle** - The Circle marks may be placed tangent to or centered on the selected object. You may specify the size (diameter) of the circle, and determine how you want to place the circle marks relative to the object. In the cases where the circle marks are tangent to the object, you may position the circle on either side of the object. The default case for placing the circle marks is Middle.
- **Block** - The insertion of a given block is made at each measuring point. To start the process, you will be asked to enter the name of the block first, and then to decide whether to align the insertion of the block with the object or not. If the block instance is to be aligned with the object, additional prompt will be given and you may specify a scale factor for the block instance as well as the type of alignment. See Special Note section.

The measuring starts at the nearest end point from the selected point on the object. For a circle entity, the measuring always starts from the zero angle point and continue in the counter-clockwise direction.

The sub-command options for the mark selections are as below:

**B** - Block, request to insert a block mark

**L** - Line, request to insert a line mark

**C** - Circle, request to insert a circle mark

The sub-command options for the placement of line/circle marks on the selected objects are as below:

**L** - Leftside, request to insert the marks on the "left" side of the object

**R** - Rightside, request to insert the marks on the "right" side of the object

**M** - Middle, request to insert the marks of which the mid-points or the center points fall on the object

Note: The determination on left or right side of the object depends on the direction of traveling along the object. The direction of traveling always starts from the end point which is closer to the picked point, to the other end point.

Only Lines, Arcs, Ellipse, Circles and Polylines are valid for this command operation. See also **DIVIDE** command.

## Special Notes

The alignment of block instance with respect to the selected object has been modified since V2.10, which clearly defines the direction of alignment. Now, if the block instance is to be aligned with the object, additional prompt will be given:

Leftside/Rightside/Middle/<Size:1.>:

You may specify the type of alignment by entering the option character "**L**", "**R**" or "**M**", respectively. The system will re-prompt again, and you may specify a scale factor for the block instance to start the operation.

The Left-side option indicates that the alignment vector points to the left side of the object along the measuring direction. The Right-side option indicates that the vector points to the right. And the Middle option, which is the default, aligns the vector with the object's traveling direction. See the figure in **DIVIDE** command.

## Procedure:

Enter the **MEASURE** command to the system command prompt:

@CMD: **MEASURE** (return)

and the system will prompt

Select object to measure:

for you to pick up the object to measure. See example procedures below.

- **To measure using POINT as marker**

Select object to measure: (pick one)

Block/Line/Circle/<Segment length>: (value)

- **To measure using Block Instance as marker**

Select object to measure: (pick one)

Block/Line/Circle/<Segment length>: **B**

Block name to insert: (select a block from pop-up window)

Align block with object : (Y or N)

Segment length: (value)

- **To measure using LINE/CIRCLE as marker**

Select object to measure: (pick one)

Block/Line/Circle/<Segment length>: **L**

Leftside/Rightside/Middle/<size (len)>: (L, R or M)

Leftside/Rightside/Middle/<size (len)>: (value)

Segment length: (value)

## Example:

**'MEM**

## Report System Memory Usage

---

**Purpose:**

The **MEM** command is used to check the system memory usage.

**Description:****DOS Specific**

The **MEM** command lets you check the system memory usage. From the report produced by the **MEM** command, you can see who is using the system memory and how much memory resource is left. This command is provided mainly for diagnostic purpose.

The **MEM** command reports the memory usage by traveling the DOS's MCB chain, which supports only the first 640KB memory onboard. Memory usage other than the 640KB DOS memory, such as HMA, UMB, and extended memory will not be reported in current release.

**Windows Specific**

A sub-command option "Resize" is added to support dynamic resizing of specific virtual memory control block.

You may directly assign the virtual memory allocation for certain control block via TwinCAD initial configuration file (TWINCAD.INI) as below:

1. Add [Memory Usage] section in TWINCAD.INI file.
2. Assign memory allocation for control block.

**BlockID = Status, Current**

BlockID: Name of virtual memory control block

Status: 0, use system default size

1, use current setting

Current: current memory setting

You may issue **MEM/M** command to inquire for the control block ID names.

**Procedure:**

@CMD: **MEM** (*return*)

MEM -- Map/Resize/<brief>: **M** (*return*)

(A list of Resizable Virtual Memory Control Blocks Current in Use is generated)

**MENU**

## Load New Menu Setup Command

---

**Purpose:**

The **MENU** command is used to load a new menu setup from disk.

**Description:**

The **MENU** command lets you change all the menu contents by loading a new one from the disk files. It will pop up a file window for you to select the menu file.

Note that a menu file, containing the definitions to screen menu, pull down menu and function keys, is not necessarily prepared in English or the language you understand. It may be in Chinese, Korean, Thai, or others. If you accidentally have these menu files and load any one of them, you will probably be confused.

You may prepare the menu file of your own. See Appendix for details about how to customize the menu file.

**Procedure:**

To load a menu file, type **MENU** command to the prompt, as below:

```
@CMD: MENU (return)
```

The system will pop up a file window, and you select the menu file to load from the window.

**Example:**

# MINSERT

## Create Array of Block Instances Command

---

### Purpose:

The **MINSERT** command is used to create an array of multiple block instances.

### Description:

The **MINSERT** command lets you create an array of block instances. It provides a very similar function to the **INSERT** command in creating block instances. However, instead of inserting only a single block instance, **MINSERT** will insert multiple instances of the block in a rectangular or linear array pattern.

You can take this command as a combination of **INSERT** and **ARRAY** commands. The only difference is that the block instances thus created are grouped as one single composite entity. Informations related to the rectangular patterns are part of this entity, instead of being a temporary specification of the multiple copies.

When you enter **MINSERT** command, the system will prompt you to specify the name of the Block to insert, the base point, the scale factor and the rotation angle of the insertion. The procedure is quite the same as that for the **INSERT** command. See **INSERT** command for details.

After you have completed specifying a single block instance, the system will start asking you about the geometries of the array pattern as it does in the **ARRAY** command under the rectangular pattern option. See **ARRAY** command for details.

After giving all the above data, you will be asked to provide one more input by the prompt:

Array base angle <rot>:

which requires you to specify the angle of the array with respect to the X-axis. If you reply with a null return, the angle will assume a default value which is the same as the rotation angle of the block instance. This is compatible with ACAD. However, if you specify a different array angle from the default value, then compatibility problem may occur, and the **DXFOUT** command will explode the array into single block instances for the output.

The rotation angle specified for a block instance can be given in absolute angle value, or in relative angle value (in **@angle** format) with respect to the array base angle. The dragging image of the array of block instances will be affected by this angle specification. If the element angle is given relative to the array base angle, the dragging image will be fully expanded, and each element will rotate about the insertion base as the array base angle changes. However, if the element angle is given in absolute mode, the image of each element block instance will be replaced with a Cross-mark (X) to notify that the element will not rotate with the array base angle during the dragging.

### Special Note

The array information of a multiple block instance supported by **TwinCAD** drawing database may contain more parameters than those provided by the **MINSERT**. Some additional parameters are set to default values for compatibility reason when you use **MINSERT** to create such block instance. However, some **TCAM** application products may need to produce Multiple block instances with these parameters set in different values. Such block instances may not be properly output via DXF file format, unless they are exploded to single block

instances. You may explode a multiple block instances into single ones, using the **EXPLODE** command.

It is the shortcoming of DXF file format that it can not convey polar array information as well as other type of grouping information for multiple block instances.

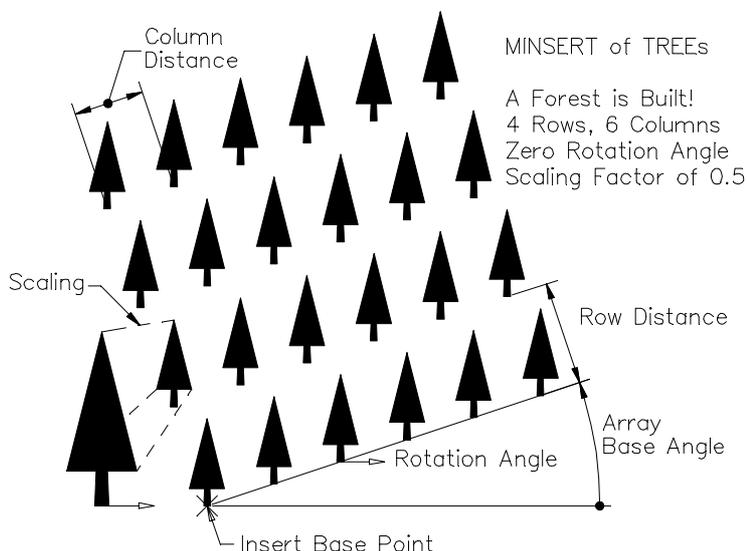
**Procedure:**

To make a multiple insert of TESTBLOCK, an example procedure is given below: (See INSERT command section for more example procedures)

```
@CMD: MINSERT (return)
Block name or (?) <...>: TESTBLOCK(return)
Insert base point: (point)
X scale factor <1>: (value)
Y scale factor <default=X>: (value)
Rotation angle <0>: (rotvalue)
Number of rows (---) <1>: (Value)
Number of columns (|||) <1>: (Value)
Unit cell or distance between rows (---): (Value)
Distance between columns : (Value)
Array base angle <rotvalue>: (return)
```

Note that if either value of the row and column number is less than 1, the command quits. If equal to 1, the corresponding question for the distance in between will not be prompted. If both are equal to 1, the insertion of block instance will become a single block instance as that done by **INSERT** command.

**Example:**



# MIRROR

## Create Mirrored Objects Command

### Purpose:

The **MIRROR** command is used to create mirrored objects.

### Description:

The **MIRROR** command lets you create mirrored images of selected objects with respect to a reference line in your drawing. After the mirror operation, you may choose to erase the original objects.

Firstly, you will need to select the objects to mirror through the object selection operation. Then the system will ask you to specify two points which define a reference axis for mirroring. When the first point is picked, the image dragging function is enabled until you enter the second point. Finally, you will be asked either to erase or to keep the original objects.

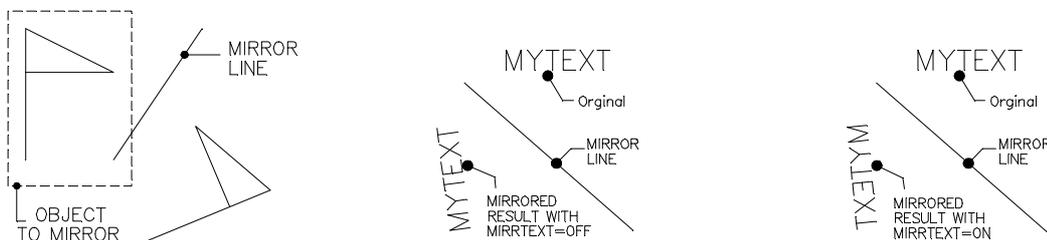
Note that the TEXT entities will be mirrored if and only if the system variable **MIRRTTEXT** is ON. Otherwise, only the location of the TEXT is mirrored and the TEXT is adjusted such that the generation of the TEXT is about at the mirrored location. This adjustment is slightly different from that by AutoCAD, but is more consistent and smooth in the transformation. Note that the TEXT in a block instance will be mirrored if the block instance is mirrored, regardless of the current setting of **MIRRTTEXT**.

### Procedure:

To create mirror objects, type **MIRROR** to the command prompt, as in the example procedure below:

```
@CMD: MIRROR (return)
Select Objects (+): (do so)
First point of mirror line: (point)
Second point of mirror line: (point)
Delete old objects? <N>: (Y or N)
```

### Example:



# MOFFSET

## Multiple Offsetting Objects Command

---

### Purpose:

The **MOFFSET** command is used to create multiple objects parallel to a specific one.

### Description:

The **MOFFSET** command lets you create large quantity of parallel objects in an efficient way. At first, you will be asked to select a base object from which multiple parallel objects are to be produced. The system will then prompt you to indicate a positive direction to offset with respect to the selected object.

After that, the system will ask you to specify the parallel objects by

- **Designating a point** through which a parallel object is to pass. The **TAN/PER** snap functions can be used when appropriate.
- **Entering a value** as the distance with which the parallel object is to offset from the base one. Positive value indicates the positive direction, and negative value, the negative direction.
- **Entering a relative value** (@value) with which the parallel object is to offset from the last created one.

You may undo the last created parallel object by entering the sub-command option "**U**". To end the operation, reply the prompt with a null return.

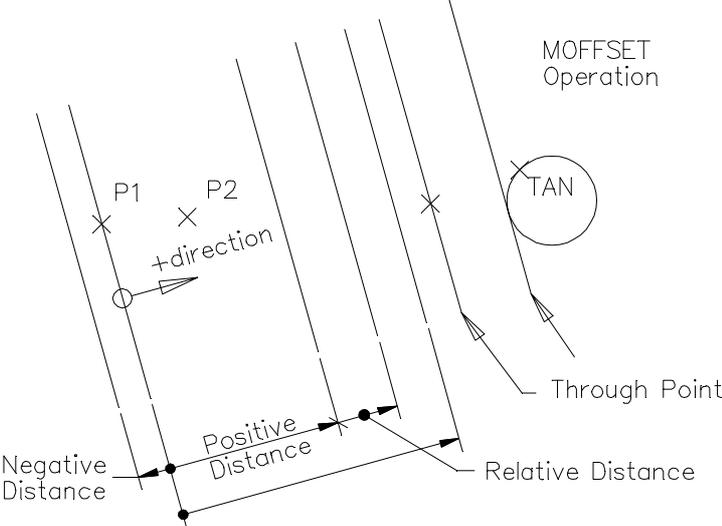
Note: Only 2D-lines, arcs, circles and polylines can be selected as the base objects in this command. And, the offsetting operation is affected by the current PUCS-plane setting.

### Procedure:

To produce multiple parallel objects, issue **MOFFSET** as in the example procedure below:

```
@CMD: MOFFSET (return)
Select offset base entity: (pick the object)
Indicate positive direction: (point)
MOFFSET -- Undo/<Offset distance: nnn>: (point or value)
...
MOFFSET -- Undo/<Offset distance: nnn>: (point or value)
MOFFSET -- Undo/<Offset distance: nnn>: (space bar)
```

**Example:**



# MOVE

## Move Objects to New Location Command

---

### Purpose:

The **MOVE** command is used to move selected objects to a new location in the drawing.

### Description:

The **MOVE** command lets you move selected objects from one location in the drawing to another.

You will be asked first to select the objects to move (see **SELECT** command for details), and then to specify the base point (a reference point from these objects) for the movement. After that, you may drag the objects to anywhere in the drawing, and finally place down the objects by designating the point desired.

The movement of the selected objects can be specified by directly entering relative displacement (**@len<ang** or **@dx,dy**) to the prompt asking for base point, followed by a null return to the next prompt.

When too many objects are selected to move, the dragging may become slow. In this case, you may turn off the dragging function using **DRAGMODE** command.

Note: During **MOVE** operation, the selected entities are moved through geometry data change, and not through copying and deleting. This leaves the generative ordering of the selected objects in the drawing database unchanged.

### Move Entities in 3-D Space

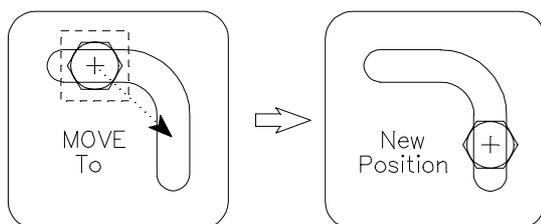
If the system variable **DO3DMODE** is 0 (OFF), which is the default, the **MOVE** command will move the selected entities in 2-D mode. That is to say, the Z-axis offset is always zero. However, if the system variable **DO3DMODE** is 1 (ON), the **MOVE** command will read the Z-component in the point designation, and use it in the moving operation.

### Procedure:

To move selected objects, type **MOVE** command and follow the procedure below:

- @CMD: **MOVE** (*return*)
- Select Objects (+): (*do so*)
- Base point or Displacement: (*point*)
- Second point of displacement: (*point*)

### Example:



# MSLIDE

## Make Slide File Command

---

### Purpose:

The **MSLIDE** command is used to produce a slide file from current view of the drawing on the graphic display.

### Description:

The **MSLIDE** command lets you save a particular view of the drawing on the graphic display to a disk file, so that the view can be retrieved later for reviewing purpose. The disk file which contains the vector image of the view, is called the slide file, since it is only a picture of the view on the graphic display.

The format of this slide file is compatible with AutoCAD's Slide file format with a slight modification for solid color fill over an irregular shape of area. However, if the view of the drawing does not contain any such solid color fill of irregular shapes, the slide file thus produced should be properly read in by other software packages that support the Slide format.

The system does not produce the slide file directly from the drawing. The slide file is produced by converting the graphic display lists from the graphic driver, **TCAM Graphic Runtime**, into the required vector format. An additional driver overlay named "**ACADSLD.OVL**" will be loaded in to do the job. The driver overlay must exist in the current logged disk, or in the data path that the the system knows.

When you issue the **MSLIDE** command, the system will pop up the file window for you to specify the output slide file. The extension of this slide file is always **".SLD"**.

### Procedure:

@CMD: **MSLIDE** (*return*)  
(*Specify slide file on file window*)

### Example:

# MULTIPLE

## Auto-repeat Next Command

---

### Purpose:

The **MULTIPLE** command is used to make the next command repeat its executions automatically.

### Description:

The **MULTIPLE** command is used to tell the system to repeat the next command entered, until it is canceled by a **<Ctrl/C>**.

Most of the commands terminate and go back to the main command prompt, as the operations are completed. To repeat such commands, you must press the return key to the command prompt without giving anything. However, you can modify the behavior of such commands by the **MULTIPLE** command. Add **MULTIPLE** command before executing the command, and that command will repeat itself after its natural termination, until you abort it by pressing **<Ctrl/C>**.

Note that some commands that pop up dialogue window like **DMODE** will never be repeated by this command.

### Procedure:

@CMD: **MULTIPLE MYCMD** (*return*)

### Example:

# MVCOPY

## Map and Copy the 3-D View of Selected Objects Command

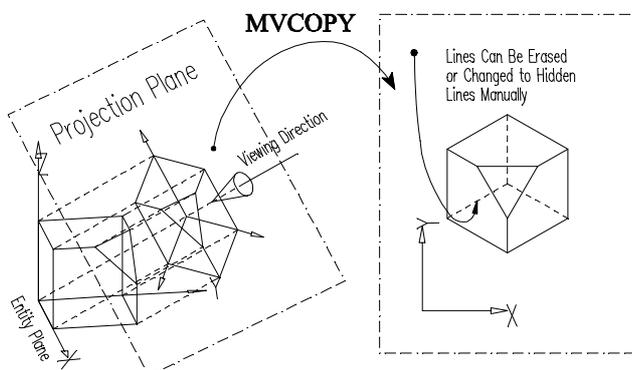
### Purpose:

The **MVCOPY** command is used to generate the 3-D view of selected objects with real 2-D entities in the drawing.

### Description:

The **MVCOPY** command lets you map a specific 3-D view of selected objects to a 2-D plane and then generate the mapping with real 2-D entities at a designated location in the drawing. It is used as a tool that provides an affine transformation operation specific for axonometric projection. It helps you in preparing the Axonometric Projection Drawings which include the Isometric, the Dimetric and the Trimetric drawings.

The operation procedure is about the same as the **COPY** command. You select those objects to copy, and then designate a base point, and an insert point (with which the first base point must coincide after being copied). However, for **MVCOPY** command, before indicating the base point and the insert point, you must set up the projection requirement, so that the system may perform the transformation calculation while copying the selected objects. The setup of the projection requirement includes the setup of the projection plane and the entity plane.



### The Projection Plane Setup

When you enter the **MVCOPY** command, you will be asked to select the objects to process via the Object Selection Operation. After that, the system will ask you to set up the projection plane with the prompt as

Rotate/<Projecting Plane (View point:<1.,1.,1.>):

or

Axoplane/Rotate/<Projecting Plane (View point:<1.,1.,1.>):

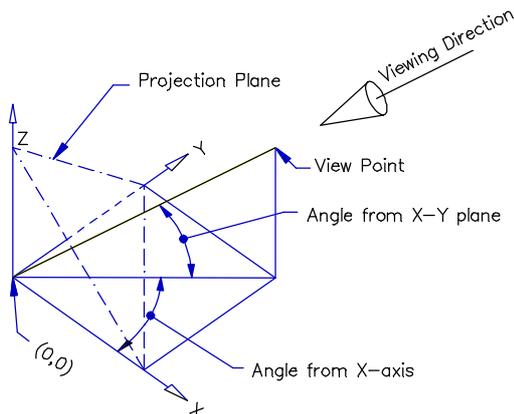
depending on whether a PUCS-plane setup is active or not.

### Projection Plane by View Point

You may enter a 3-D view point from which the viewing direction is taken toward the drawing origin. The vector from the origin to this 3-D view point will be a normal vector to the projection plane.

The following is a table showing the correspondence between the normal vectors of the projection planes and the type of axonometric projection drawings.

$(\pm 1, \pm 1, \pm 1)$	Isometric Projection
$(\pm 1, \pm 1, N)$	Dimetric/Top Projection
$(N, \pm 1, \pm 1)$	Dimetric/Left Projection
$(\pm 1, N, \pm 1)$	Dimetric/Right Projection
$(N_1, N_2, N_3)$	Trimetric Projection.



### Projection Plane by Rotation Angle

You may specify the projection plane via the axis rotation method, by entering the sub-command option "R". Using this method, you will be asked to define the viewing direction by an angle from the X-axis on the X-Y plane, and another angle toward the Z-axis away from the X-Y plane. The system will give in sequence the following prompts:

```
Enter angle in X-Y plane from X axis <45.°>:
Enter angle from X-Y plane <35.264°>:
```

The equivalent view point will be calculated as  $(\text{Cos}(A), \text{Sin}(A), \text{Tan}(B))$ , where A is the angle of the vector projected on the X-Y plane, and B, the angle between the viewing direction from the X-Y plane.

### Projection Plane by Current PUCS-plane Setup

If the current PUCS-plane is active and valid for an axonometric projection setup, you may enter the sub-command option "A" to specify to use the same setup for the projection plane, instead of giving the viewing direction. This is the most easy way to define the projection plane. The MVCOPY command will establish the transformation matrix based on the axes setup. You may use the PUCS, AXOPLANE and ISOPLANE commands to define the required projection plane.

Note: The Oblique Projection can not be produced by MVCOPY command, so the axes setup for Oblique Projection will not be acknowledged for a valid PUCS-plane setup.

### Orientation of the Projection Plane

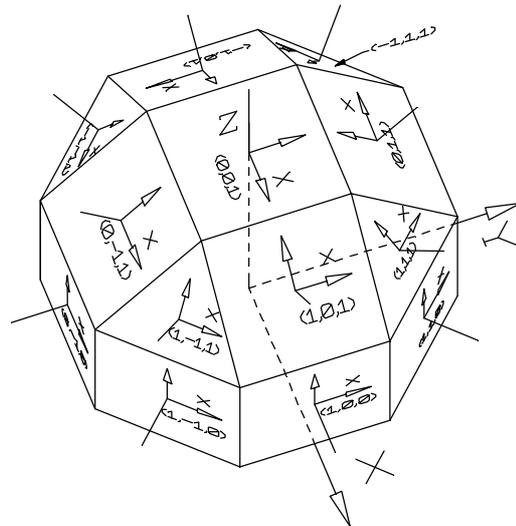
If the projection plane is defined by a normal vector to the plane, it defines only the viewing direction. Assuming that a semi-transparent paper is laid on the projection plane and the projected image is depicted on that paper, you will find that there are many ways to look at

this paper -- you may rotate the paper to any orientation you like, to observe the projected image. A complete specification of the projection plane must thus include the orientation of the "paper".

To simplify the operation, a specific orientation of the "paper" unique to each projection plane is therefore determined by the system automatically according to the drawing convention, which always let the projected Z-axis in the vertical direction. This convention conforms with our experience in observing objects when we stand or sit erect to see.

However, if the projection plane is defined by the current PUCS-plane setup that conforms with an Axonometric Projection, the orientation of the "paper" is also given. The projected Z-axis is then not necessarily in the vertical direction.

You may use both methods to define the same projection plane, but that does not mean the two methods are totally equivalent.



## The Entity Plane Setup

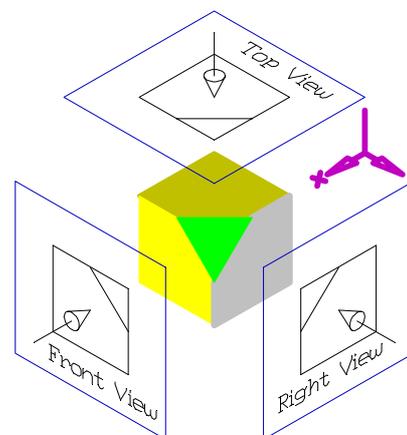
After setting up the projection plane, the system will continue to ask you to set up the entity plane, with the prompt:

```
Top/Right/Front/<Entity Plane:(0.,0.,1.)>:
```

The entity plane defines the coordinate plane at which the selected entities will be assumed to be placed for the projection calculation. You may draw a box on the X-Y plane of the model space, and tell **MVCOPY** that the box will be placed at the Y/Z plane or a skew plane for the projection view.

You may directly input the direction vector normal to the entity plane, or specify the plane by the following sub-command option:

- RO** Rotate, request to specify the plane by the axis rotation method, the same as that for the projection plane.
- T** Top, specifying the X-Y plane as the entity plane. The direction vector normal to this plane is (0,0,1). This is also equivalent to specify the top view of the drawing, if the selected objects represent a view of the orthographic drawings.
- R** Right, specifying the Z-X plane as the entity plane. The direction vector normal to this plane is (0,1,0). This is also equivalent to specify the right-side view of the drawing, if the selected objects represent a view of the orthographic drawings.



- F** Front, specifying the Y-Z plane as the entity plane. The direction vector normal to this plane is (1,0,0). This is also equivalent to specify the front view of the drawing, if the selected objects represent a view of the orthographic drawings.

### Coordinate System on the Entity Plane

When an entity plane is defined (by a normal vector), a coordinate system will be uniquely determined on the plane, so that the overall specification for the projection requirement can be simplified.

The determination of the coordinate system on the Entity Plane follows the same rule as that on the Projection Plane. It assumes that this coordinate system's origin is at (0,0,0) of the model space and its Y-Z plane will contain the Z axis of the model space. This means that, if we are looking down to the entity plane from its +Z axis toward its origin, letting coordinate system's +X axis horizontally point to the right and its +Y axis vertically point to the top, we will find that the Z axis of the model space (projecting on to this viewing plane) coincides with its Y axis or degenerated to a point at the origin (as a special case).

There exists two special cases, the **(0,0,1)** and **(0,0,-1)** planes. The (0,0,1) plane will be the same as the X-Y plane of the model space, so its coordinate system will coincide with the world coordinate system (**WCS**). The (0,0,-1) plane is also coincident with the X-0 plane of the model space. However, as its +Z axis is pointing to the same direction as the -Z axis of WCS, its +X axis and +Y axis direction can not be the same as the WCS (remember that we are using a right-handed coordinate system). It is so chosen that the +X axis direction will be the same as the -Y axis direction of the WCS, and the +Y axis direction, the same as the -X axis direction of the WCS.

### Placement of the Mapping Result

After you have specified the projection requirement, the system will prompt

Base point:

asking you to designate a 3-D point as a base reference point on the Entity Plane. You may snap the point from the selected objects, using the snap directives. If you input a point coordinate without the z component, the current default elevation will be used.

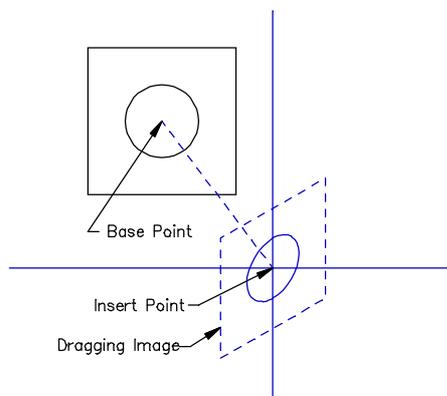
The image of the projection result will be generated and the dragging of this image will be started as soon as the system prompts

Insert base point:

asking you to designate a 2-D point on current modal space where to put the projection result. It indicates where the base reference point should be placed after the transformation. The dragging image thus serves as a preview of the mapping production, and is very convenient for you to justify the projection requirement as well as placement.

During the dragging, if you change the content of the system variable **MVSCALE** transparently, the size of the dragging image will be dynamically changed too. Of course, if you change the size of the drawing window, the dragging image will be re-sized accordingly.

Note: If the projection plane is set up by the "A" option (to use the current PUCS-plane setup), you may use the <Ctrl/I> function to switch the current PUCS-plane setup and the dragging of the projection image will be changed accordingly. In short, you may transparently issue the



**PUCS**, **AXOPLANE**, **ISOPLANE** commands to change the PUCS-plane setup, and change the result of the dragging image dynamically.

## The Production of the Entities in Projection

After you have designated the corresponding position where to insert the projection result, the system will start to process the selected entities and create new entities to meet the projection requirement. Except the Dimension entities, all newly created entities will assume new properties from the current default setup.

All selected entities will be processed as 3-D objects. The plane entities with different elevation and extrusion thickness, or on different ECS plane, will all be processed accordingly. Plane entities with extrusion thickness will be taken as a wall. The transformation will result in two parallel profiles (showing different elevations) with additional line segments added to exhibit the thickness of objects or the object boundaries.

However, not all the entities can be processed by **MVCOPY** command. And, the transformation rules varies with the type of the entities.

**LINE** The result of the transformation of a single line entity is still a line entity. However, if the resulting line should degenerate to a point (as the line parallel with the viewing direction), it will be discarded.

**ARC** The transformation of a single arc will produce an elliptic arc, which may be represented by an open polyline comprising of smooth arc segments or by a true Ellipse-arc entity, depending on the current setting of system variable **ELLIPSEARC**. However, if the plane on which the arc resides should be parallel with the viewing direction, its transformation result will be degenerated into lines.

Also, if the arc has an extrusion thickness and the wall surface should contribute a boundary line in the projection view, additional line will be added to show the boundary line automatically.

**CIRCLE** Like the arcs, the transformation of a circle will produce an ellipse, which may be represented by a closed polyline comprising of smooth arc segments or by a true Ellipse entity, depending on the current setting of system variable **ELLIPSEARC**. However, if the plane on which the circle resides should be parallel with the viewing direction, its transformation result will be degenerated into lines. Also, if the circle has an extrusion thickness and two additional line will be added to show the boundary line of the wall surface along the viewing direction automatically.

**ELLIPSE ARC** The result of the transformation of an ellipse-arc is still an ellipse-arc. The rules for it are the same as those for the **ARC**.

**ELLIPSE** The result of the transformation of an ellipse is still an ellipse. The rules for it are the same as those for the **CIRCLE**.

**POLYLINE** A polyline is composed of line, arc and ellipse-arc segments. So, the transformation of a polyline will be processed by its comprising segments one by one. However, the redundant boundary lines between two consecutive segments will be removed automatically. The original polyline relationship between converted segments will be preserved after the

transformation, and so are the properties of FILLSOLID, CLOSED of a closed polyline.

**TEXT**

The result of the transformation of a Text entity is still a copy of the Text entity. It is the control data of the text entity, such as the oblique angle, width factor, text height and character distance, being modified to exhibit the projection effect. The original text justification requirement will be preserved after the transformation. The only case that the transformation may not produce the true projection view of a text entity is when the text is in circular writing, since the system does not support 'elliptic' text. The center position will be determined for the best approximation, yet the quality can not be guaranteed.

**SYMBOL**

The result of the transformation of a Symbol entity is still a copy of the Symbol entity. However, the control data of the symbol, such as the X-Y scale factor and Oblique angle, will be modified to exhibit the projection effect.

**3DFACE**

The visible edges of the 3DFACE entity will be transformed as the line entities.

**DIMENSION**

The result of the transformation of a Dimension entity may still be a single Dimension entity or a group of drawing representing the projection result, depending on the type of the Dimension entity. The original properties of the dimension entity (such as layer and color) will be inherited by the transformation result.

- **Linear Dimension Entity** -- The result of the transformation is still a Linear Dimension entity with proper Rotated and Oblique specification. All local dimension parameter data are transformed accordingly. However, as the result is still represented by the Dimension entity, it will be affected by the global control data, such as dimension text width factor. The dimension text will be larger than it should be. You may use **SETDIM** and **CHANGE** command to modify such entity after the transformation.
- **Angular Dimension Entity** -- The result of the transformation is a group of drawing comprising lines, arcs and texts, as if you have exploded the Dimension entity before doing the transformation.
- **Diameter/Radius Dimension Entity** -- Ditto.
- **Ordinate Dimension Entity** -- Ditto.
- **Leader Dimension Entity** -- The result of the transformation is still a Leader Dimension entity. However, it is the node points of the dimension being transformed. The shape of the arrow pointer can not be transformed to have the projection effect. To have true projection effect of the arrow pointer, you may explode the dimension before doing the transformation.
- **Center Dimension Entity** -- The result of the transformation is still a Center Dimension entity with oblique/scaled feature to show the true effect of the projection.

## Viewing Direction vs. PUCS-plane Axes Setup

You may set up the projection plane by specifying the viewing direction (plane normal vector) or by using the current PUCS-plane axes setup. However, these two methods are not totally equivalent. They both have their own advantage and limitations which will be discussed briefly in the following paragraphs.

### Viewing Direction Method

The viewing direction method specifies a fixed direction to view the objects. It clearly defines the topological relationship among objects in terms of what can be seen and what should be hidden. The transformation matrix can be set up for all the possible views, even though it always assumes that the projected Z-axis be in the vertical direction.

The only disadvantage is that it is not easy for a user to determine the required viewing direction for a specific kind of projection drawing requirement. However, tabulation of commonly used view points should help to solve this problem.

### PUCS-plane Axes Method

On the other hand, since the PUCS-plane axes setup has already defined the projected axis directions for an axonometric projection, a transformation matrix that may produce the same projected result can be found for the **MVCOPY** command to work. You may easily set up the PUCS-plane for a specific kind of projection drawing requirement by using the **PUCS**, **AXOPLANE** and **ISOPLANE** commands. You may also define the orientation of the projection plane, using this PUCS-plane method.

However, for a set of PUCS-plane axes setup, there is no unique correspondence to the viewing direction. For example, the (1,1,1) and (-1,-1,-1) viewing directions will produce the same PUCS-plane axes setups in different orientations (image different only in 2-D rotation on the projection plane). Given the specific set of PUCS-plane axes setup, the viewing direction must be chosen from either one. It is apparent that this method does not cover all the possible views.

The viewing direction is important in the determination of the topological relationship among objects for hidden lines removal, although the hidden line removal is not yet supported in current version.

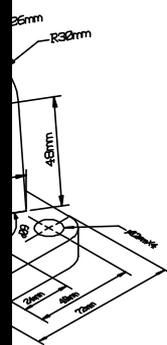
### Special Note

Although 3DFACE entities and objects with extrusion thickness are processed by this **MVCOPY** command, the hidden-line removal capability is not yet provided in current release. This is because the assumption that the extrusion thickness can be a wall surface to hide objects in behind does not make much help in preparing the Axonometric Projection drawing. You may, however, edit the resulted drawing to meet your own requirement easily.

### Procedure:

-

Ports are created  
of TCL functions  
of out\_obj() with



**NEW**

## Set New Drawing Command

---

### Purpose:

The **NEW** command is used to set new work drawing.

### Description:

The **NEW** command lets you set new work drawing without exiting **TwinCAD**. There are three options:

- L**     **Load new work**, request to abandon current work drawing and load a new work drawing. The newly loaded work drawing will become current work drawing. You will be asked for the name of work drawing to load in. See also **LOAD** command for drawing loading details.
- C**     **Create new work**, request to abandon current work drawing and clear everything for a totally new work drawing. **TwinCAD** will automatically load in the default prototype drawing and then ask you to specify the name of the new drawing to create.
- S**     **Set output filename**, request to set the output filename of the current work drawing, or to rename current work drawing to a new work drawing. You will be asked for the name of the new drawing to become.

If you press the space bar to return a null reply to the prompt, the system will do nothing but quit the command immediately.

### Special Notes

- Once this command is successfully executed, the **UNDO** and **RECORD** buffer will be cleared.
- You may specify to load/create DWG file, provided that the file extension "DWG" is explicitly given.
- Whenever a new drawing file is loaded via **NEW/Load** command, **TwinCAD** will automatically execute the extended command "WRKSTART", provided that the command is well defined (loaded in the system).

### Procedure:

@CMD: **NEW**

Warning! Undo/Record buffer will be cleared.

New -- Load-new-work/Create-new-work/Set-output-filename/<Nothing>:

### Example:



**ODIM**

## General Ordinate Dimension Command

---

**Purpose:**

The **ODIM** command is used to create ordinate dimensions.

**Description:**

The **ODIM** command lets you create ordinate dimensions. It is the formal entry command for the ordinate dimension commands: **XDIM** and **YDIM**. See the description of these commands for details.

**Procedure:**

Enter the **ODIM** command to the system command prompt:

@CMD: **ODIM** *(return)*

and the system will prompt, .

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>:

if the last ordinate dimensioning is for X-coordinate. If the last ordinate dimension mode is for Y-coordinate, then the prompt will be:

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>:

showing that you are in Y-type ordinate dimensioning.

**Example:**

# OFFSET

## Create Parallel (Offset Compensated) Objects Command

### Purpose:

The **OFFSET** command is used to create a parallel object to a selected object.

### Description:

The **OFFSET** command lets you create an object that is 'parallel' to a specific object at a given distance or through a specified point. The word 'parallel' means, the shortest distance from every point of the generated object to the selected object remains constant. This definition is quite straightforward and obvious for the primary entities like lines, arcs and circles. As for a composite entity like polyline, however, this definition needs to be elaborated and will be explained in later paragraphs.

When you enter the **OFFSET** command, you will be asked first to select the object to offset or to select the method to offset, as the system prompts

Select object to offset <xx>/Through:

where xx is the last offset specification given to **OFFSET** command. The system provides two kinds of offset specifications:

- **Through point**, specified by the sub-command option "T", requesting to produce an offset object passing through a designated point. You will be asked to designate the point to pass through each time you select a new object to offset. You may designate the through point by the **PER/TAN** object snap directives if appropriate.
- **Offset distance**, specified by entering a distance value, requesting to produce an offset object at the specified distance from the selected one. You will be asked to indicate the side to offset each time you select a new object to offset.

If you change the offset specification by entering new offset distance or the sub-command option "T", the system will update the message and prompt again. You must pick up an object to proceed the offset operation, and the system will prompt the message again after an offset operation is completed.

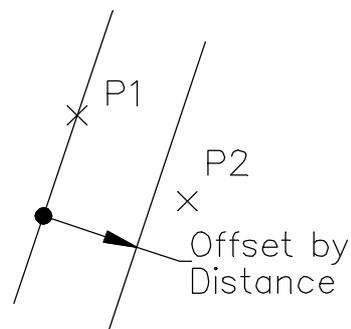
To exit the command, reply to the prompt with a null return or type a **Ctrl/C**.

### Offset by Distance

After you have picked up an object, with an offset distance given, the system will prompt:

Side to offset:

asking you to indicate which side of the object to create its offset. For example, if the entity you select is an ARC, and the offset to create is inside of the ARC, then you must designate a point within the corresponding circle of the ARC.

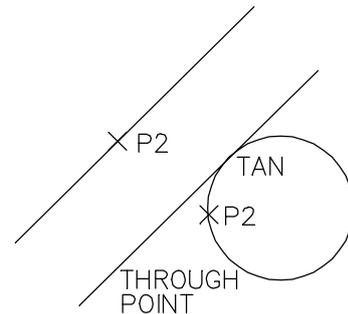


## Offset by Through Point

If the *Through-point* mode is enabled, after you have picked up an object, the system will prompt:

Through point:

asking you to designate the point through which the offset object will be created. The object snap directive **TAN** can be used to specify the offset object to be tangent to a given line/arc/circle.



## Error Messages

If the effective offset distance is zero (either explicitly given or calculated under the through-point mode), the system will issue the error message:

Invalid through point or zero offset.

and ignore that offset operation. If the **OFFSET** command can not find a proper offset to the selected object, a general error message:

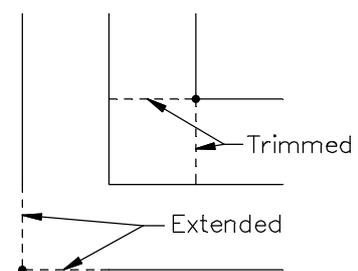
Can't have offset.

will be given. Examples are when you have selected an entity other than line/arc/circle/polyline, or when you specify the offset of a circle which will produce a negative radius. Moreover, if you select an arc, a circle or a polyline containing arc segment when the PUCS-plane axes setup is active, this message will also be given.

## Offsetting a Polyline

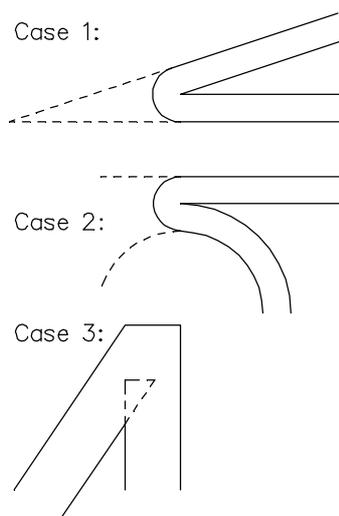
Since a polyline is composed of successive segments of lines and/or arcs, which represents an arbitrary curve, the 'parallel' curve to a polyline can not be well defined. If the definition of two 'parallel' objects is that the shortest distance between the two objects remains constant everywhere, we must further define the shortest distance measured between two 'parallel' polylines.

The shortest distance taken from a point on one polyline to another 'parallel' one, should be the distance measured from that point to the corresponding segment on the 'parallel' polyline. If the point is a common point to two adjacent segments, then the shortest distances taken from the point on both segments should be the same. The 'parallel' polyline is therefore created by offsetting each segment from the original polyline and then trimming over or extending any two adjacent offset segments to their points of intersection where they can be joined together to form a continuous polyline.



However, in practice, the system has made some changes in the method to find a 'parallel' polyline, to avoid the possible fault cases:

- Case 1: Sharp corner tends to extend out more than acceptable, when the offset path is made outside of the corner.
- Case 2: No intersection can be found between two adjacent offset segments, or the offset segment does not exist (negative radius).
- Case 3: The direction of an offset segment is reversed and thus the offset path is twisted about the segment.

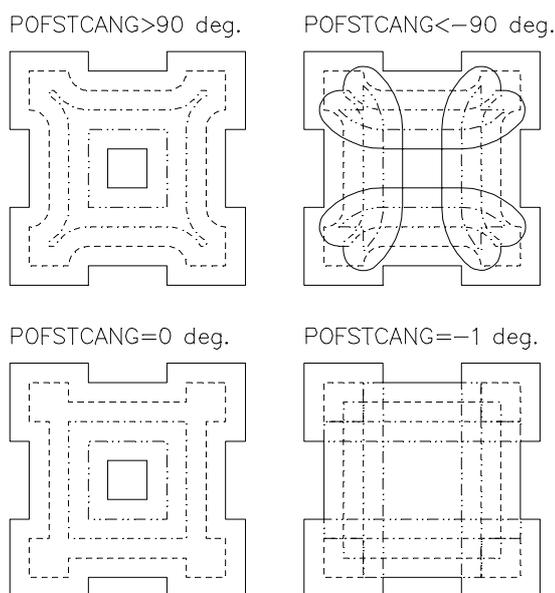


In **TwinCAD**, the offset path of a polyline is generated like a locus formed by the center of a rolling circle. The circle has the given offset distance as its radius, and will generate the offset path by rolling along the specified polyline on the side indicated. For the path generated, the original sharp corners will be changed to circular arcs without the need of finding the intersection points. Using the method as described above, the first two fault cases can be avoided. And as a more sophisticated method is applied, the third case can also be avoided to a certain extent.

The rounding effect at the outer corner is controlled by the system variable **POFSTCANG**, which specifies the minimum corner angle that won't get the rounding treatment. All the corner angles that are smaller than the absolute value of **POFSTCANG** will be rounded as mentioned earlier. Setting **POFSTCANG** to zero will completely disable the rounding effect, since no corner angles can be smaller than this.

You may further disable the internal function that tries to resolve the possible undercut problem (the third case from the above), by setting the value of **POFSTCANG** to a negative value. That is, the polyline offset will be produced in plain mode, without any advanced trimming consideration over the offset segments. The only trimming that will be done is to make adjacent segments remain joined together (as a nature of polyline). You may trim the resulted polyline by yourself.

Note: An additional arc segment will always be added if no intersection can be found between two adjacent offset segments, despite of the setting value of the **POFSTCANG**.



## Through Point Consideration

There are quite often an infinite number of 'parallel' curve passing through the same point, to a given polyline. If a through point is given to find the offset path, the system will search for the one that has the minimum offset distance.

## Offsetting a True Ellipse Entity

If the PUCS-plane axes setup is not active, the **OFFSET** command will offset the ellipse or elliptic-arc by adding/subtracting the offset amount to/from the length of its both major and minor axes. However, the resulted ellipse is not the true 'parallel' curve to the selected ellipse, since the 'parallel' curve of an ellipse is NOT AN ELLIPSE!

If you are to find the true 'parallel' curve to an ellipse, you should explode it first to the desired precision (as being controlled by the system variable **SPLINESEG**) and then offset the resulted curve.

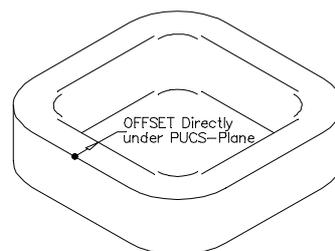
## Offsetting Objects on PUCS-Plane

The **OFFSET** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to offset lines, ellipses and polylines on the projected plane from the virtual space directly, as described below:

**LINE** The line will be transformed into the virtual space to find its offset which is then transformed back as a result.

**ELLIPSE** The ellipse (ellipse arc) will be checked if it is the one that can be produced under current PUCS-plane axes setup by projecting a circle from the virtual space. If it is, the circle (found by reversing the projection) will be offset and projected again as the result of the offset operation. That is to say, the ellipse, if applicable, will be taken as a circle in the projected plane. Otherwise, no offset can be found.

**POLYLINE** If the polyline contains only line segments or ellipse-arcs that can be transformed back to the virtual space as an arc, then it will be offset with the required transformation as described in the above. Otherwise, no offset can be found.



**Others** No offset can be found.

This feature is useful in preparing the projection drawing.

## Special Notes

By default, the properties of each newly created offset entity will be directly copied from its original entity. However, you may instruct the **OFFSET** command to use the current default properties (those set up by **EMODE** command) for these newly created entities, by setting the system variable **USE\_EMODE** to 1 (ON). Affected entity properties are layer, color, linetype, elevation and extrusion thickness.

## Procedure:

- To offset objects by 10 drawing units, follow the procedure below:

@CMD: **OFFSET** (*return*)

Select object to offset <0.>/Through: **10** (*return*)

Select object to offset <10.>/Through: (*pick object*)

Side to offset: (*point*)

...

Select object to offset <10.>/Through: (*space bar*)

- **To offset objects through specific points, follow the procedure below:**

@CMD: **OFFSET** (*return*)

Select object to offset <0.>/Through: **T** (*return*)

Select object to offset <Through>/Through: (*pick object*)

Through point: (*point*)

...

Select object to offset <Through>/Through: (*space bar*)

**OOPS**

## **Restore the Last Erased Objects Command**

---

**Purpose:**

The **OOPS** command is used to restore the last erased objects.

**Description:**

The **OOPS** command lets you restore the last erased objects. This is an ACAD-compatible command. If possible, use **UNDO** command instead.

**Procedure:**

To restore the last erased objects, issue the **OOPS** command, as below:

@CMD: **OOPS** (*return*)

**Example:**

**'ORTHO**

## Orthogonal Drawing Mode Command

---

**Purpose:**

The **ORTHO** command is used to turn the orthogonal drawing mode **ON** or **OFF**.

**Description:**

The **ORTHO** command lets you turn the orthogonal drawing mode **ON** or **OFF**. When the orthogonal drawing mode is **ON**, a line that is being dragged on the screen (with one end fixed and the other end following the cursor) will be constrained to the direction of one of the coordinate axes. The status line will display "ORTHO" when this mode is ON.

This is an ACAD-compatible command. You may use **DMODE** command to set up the orthogonal drawing mode with dialogue window. Or, you may type **Ctrl/O** to toggle the orthogonal drawing mode. See also **DMODE** command.

Note: The **ORTHO** mode will be affected by the current snap axes setting. The constraining direction will be aligned with the axes specified by the system variables **SNAPANGLE** and **SNAPBETA**.

**Procedure:**

@CMD: **ORTHO** (*return*)

@CMD: **ORTHO ON/OFF**: (*ON or OFF*)

**Example:**

# 'OSNAP

## Set up Default Object Snap Directives

---

### Purpose:

The **OSNAP** command is used to set up default object snap directives.

### Description:

The **OSNAP** command lets you set up the default object snap directives. So whenever a point data input is required during the operation, the default object snap function will be applied automatically.

When you issue the **OSNAP** command, the system will ask you to enter a character string containing the object snap directives, each separated by a comma, as in the example below:

Object snap mode: **INT,END,MID** (*return*)

To remove the object snap directives, reply with a null return as in the example below:

Object snap mode: (*return*)

This is an ACAD-compatible command. You may use **DMODE** command to set up the directives in dialogue window. Note that some commands may override this OSNAP in their operations.

Valid Object Snap Directives are **ENDp**, **MID**, **CEN**, **INT**, **INS**, **PER**, **TAN**, **QUA**, **NODe**, **NONE**, and **NEAr**. Note that if **NEAr** is used with other directives, then it will have the lowest priority in returning the point. If **NONE** is also given in the string, then the cursor point will be returned if no point can be snapped by applying the other object snap directives.

### Procedure:

@CMD: **OSNAP** (*return*)

Object snap mode: **directives** (*return*)

### Example:

As long as the object snap directives are in effect, whenever you are asked to designate a data point, the system will prompt out the current effective object snap directives, as in the example given below:

@CMD: **OSNAP**

Object snap mode: **int,cen,near,non** (*return*)

@CMD: LINE (*return*)

@CMD: LINE From point: CEN, INT of, NEA to, <pnt>

The priority of data point return is in the order below:

1. Center point or Intersection point of objects that is nearest to the cursor point. If the picked objects do not contain the intended point (center or intersection point in this case), then no point will be snapped by these two directives.
2. Nearest point on the picked object to the cursor point. If no object is picked by the cursor point, then no point can be snapped by the NEAR directive.
3. If no point can be snapped by the above two cases, then the cursor point is returned.

Note that this order is not implied by the order the object snap directives were entered, but by the internal rules. For example, if the NEAR directive is not entered, then case 2 will be ignored; and if the NONE directive is not entered, then no point will be returned when the first two cases fail to return data point.

# 'PAN

## Move Current View Window Location Command

---

### Purpose:

The **PAN** command is used to move the current view window to a new location.

### Description:

The **PAN** command lets you move the current view window to a new location without changing its size. However, as the display screen is fixed to you during operation, you may reverse the concept as to slide the drawing across a fixed view window and view different portion of it, though actually the drawing is not moved.

Therefore, you will be asked to specify the movement of the drawing relative to the view window instead of the movement of the window, by either designating two points on the screen that specify the displacement vector, or entering the displacement in terms of coordinate pairs directly. There are three ways for you to input coordinate pairs from keyboard:

- To enter two coordinate pairs in sequence, and the displacement from the first point to the second is calculated.
- To enter one single coordinate pair, followed by a null return. In this case, the coordinate entered will be taken as the relative displacement of the drawing with respect to the screen.
- To enter one single coordinate pair in relative mode, i.e., the coordinate entry is preceded with @ (at sign). In this case, the 'Second point' prompt will not appear, and the coordinate input will become the relative displacement.

### Procedure:

To view different portion of the drawing, type **PAN** command and follow the procedure below:

@CMD: **PAN** (return)

Base point or Displacement: (point)

Second point: (point)

or

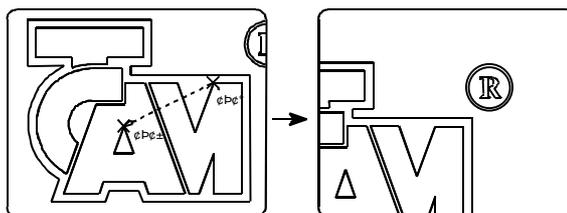
Base point or Displacement:

(coordinate pair)

Second point: (space bar)

or

Base point or Displacement: (relative coordinate pair)



### Example:

# PASTECLIP

## Paste in Drawing from Clipboard Command (Win)

---

### Purpose:

The **PASTECLIP** command is used to paste in drawing from the Windows Clipboard.

### Description:

The **PASTECLIP** command lets you paste in drawing from the Windows Clipboard, if there is clipboard data available in one of the following formats (listed in priority order):

<b>WRK Drawing</b>	<b>TwinCAD's</b> native drawing format, saved by <b>COPYCLIP</b> command.
<b>DWG Drawing</b>	<b>AutoCAD</b> R12's or R13's DWG drawing format, saved by <b>AutoCAD</b> R12/R13 <b>COPYCLIP</b> command or saved by <b>TwinCAD's</b> <b>COPYCLIP</b> command.

If there is a drawing data in supported format available from the clipboard, **PASTECLIP** will insert it into the current drawing in the same way as you insert an external drawing via the "INSERT \*=extfile" command. You will be asked to designate the base point and specify the scale factors and rotation angle of the insertion, as the prompts in sequence:

Insert base point:  
X scale factor <1>:  
Y scale factor <default=X>:  
Rotation angle <0>:

See also **INSERT** command for more details of drawing insertion.

### Special Notes

The **PASTECLIP** command is intended to paste in drawing objects. To paste in Text data, use **PASTEXT** command.

### Procedure:

@CMD: **PASTECLIP** (*return*)  
Insert base point: (*point*)  
X scale factor <1>: (*value*)  
Y scale factor <default=X>: (*value*)  
Rotation angle <0>: (*value*)

### Example:

# PASTEXT

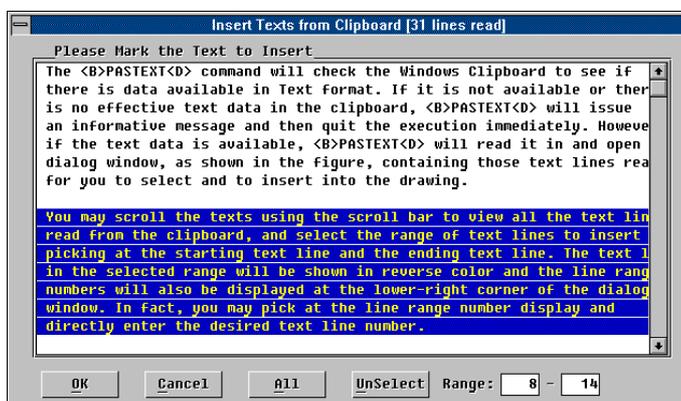
## Paste in Texts from Clipboard Command (Win/TCL)

### Purpose:

The **PASTEXT** command is used to paste in selected texts from the Windows Clipboard if the texts are available.

### Description:

The **PASTEXT** command will check the Windows Clipboard to see if there is data available in Text format. If it is not available or there is no effective text data in the clipboard, **PASTEXT** will issue an informative message and then quit the execution immediately. However, if the text data is available, **PASTEXT** will read it in and open a dialog window, as shown in the figure below, containing those text lines read for you to select and to insert into the drawing.



You may scroll the texts using the scroll bar to view all the text lines read from the clipboard, and select the range of text lines to insert by picking at the starting text line and the ending text line. The text lines in the selected range will be shown in reverse color and the line range numbers will also be displayed at the lower-right corner of the dialog window. In fact, you may pick at the line range number display and directly enter the desired text line number.

You may press the [**ALL**] button to select all the text lines, or [**UnSelect**] button to release the current range selection for the next new range selection. Pressing the right button of the mouse pointer has the same effect as pushing the [**UnSelect**] button. You may cancel the whole operation by pressing the [**Cancel**] button or pressing <ESC> key from the keyboard.

Once a valid range of text lines is being selected, you may press the [**OK**] button to proceed to the next operation step. **PASTEXT** will enter **DDSTYLE** dialog window for you to proceed the text style setting operation before inserting the texts. See **DDSTYLE** command reference for details about its operation.

If you press the [**OK**] button to terminate the **DDSTYLE** operation, **PASTEXT** will start to prompt at the command area in sequence:

Text Insert Point:  
Text Insert Angle <0.°>:

asking you to complete the text insertion by supplying the appropriate data. Once these entries are completed, **PASTEXT** will insert these selected texts in the drawing and then terminate.

Note that **PASTEXT** will limit the maximum number of lines read from the clipboard to 400 lines, and each line can contain at most 80 characters in current version. Exceeding text data will be truncated.

**Procedure:**

@CMD: **PASTEXT** (*return*)  
... [Dialog Operation] ...  
Text Insert Point: (*point*)  
Text Insert Angle <0.°>: (*value*)

**Example:**

**PATHCUT****Optional CAM package for NC Wire-Cutting Machine**

---

**Purpose:**

The **PATHCUT** command is used to access the optional CAM package for NC wire-cutting machine.

**Description:**

The **PATHCUT** command lets you prepare and make the tooling paths for NC wire-cutting machine and the likes. This is an optional package specific for NC wire-cutting machine. Detailed information regarding this command operation and other related technical information are given in a separate document.

Interested user may contact our local dealer for further information.

# PCXOUT

## Export Drawing Images to PCX Format File Command (Win)

---

### Purpose:

The **PCXOUT** command is used to export the drawing image to disk file in PCX. The image will be generated in the same way as the **PRPLOT** command prints the drawing to the printer.

### Description:

The **PCXOUT** command lets you export drawing images to disk files in the following PCX image types:

- RLE compressed monochrome image in white color background.
- RLE compressed monochrome image in black color background.
- RLE compressed 16-color image in white color background.
- RLE compressed 256-color image in white color background.

The images will be generated in the same way as you prints the drawing to the printer. However, apart from output to the printer, which has fixed resolutions and X/Y aspect ratio, printing image to bitmap file requires you to determine the output resolutions and the X/Y aspect ratio. So, the operation of the **PCXOUT** command is about the same as the **PRPLOT** command, excepts that the details of the image export configuration is different.

In fact, **PCXOUT** command works exactly the same way as **BMPOUT** command, excepts that the resulting bitmap image is converted and saved to the disk file in the required PCX file format.

See **BMPOUT** command for the operation details and special notes about the image file export.

### Procedure:

@CMD: **PCXOUT** (*return*)

What to plot -- Display, Extents, Limits or Window <D>:

...[Dialog Window Operation]...

# PJOIN

## Join Segments into A Single Polyline Command

---

### Purpose:

The **PJOIN** command is used to join selected segments into a single polyline.

### Description:

The **PJOIN** command lets you join selected segments into a single polyline and control the possible branching during the joining.

You will be asked to select the starting segment of the polyline-to-be first, and then select the segments that will become the constituents of the polyline. As a start, the first segment is converted into a polyline containing itself. If it is already a part of a polyline, then the whole polyline will be taken. **TwinCAD** then searches these selected segments for possible joining to the end of the polyline. If found, it will be added to the polyline and the searching process will start all over again, until no more segments can be found or the polyline is closed.

During the joining operation, **TwinCAD** will start searching forward and then backward for possible joining. If there are more than two segments found to join with the polyline at the same point, i.e., more than two possible branches at the point, **TwinCAD** will highlight these segments, place a mark at that point, and prompt the message:

*nn possible branches encountered, please select the desired one:*

You will have to pick up the desired one from those highlighted segments to continue the joining process.

Note that you can also select segments of open polylines to join with the new polyline, while they are taken as a whole.

After the end of the joining operation, **TwinCAD** will report the result as

*nn entities added, and polyline is closed.*

if the polyline is closed, or

*nn entities added, and polyline is still open.*

if the polyline is still open.

Use the **AUTOJOIN** command for faster joining of more than one polyline, if the branching problem is not a concern.

### Procedure:

@CMD: **PJOIN** (*return*)

Select starting segment: (*pick one*)

Select Object (+): (*do so*)

...

### Example:

**'PLAN****Set to 2D Plan View Command**

---

**Purpose:**

The **PLAN** command is used to set the current view point to a 2D plan view.

**Description:**

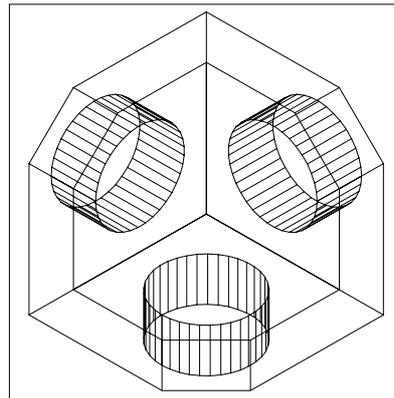
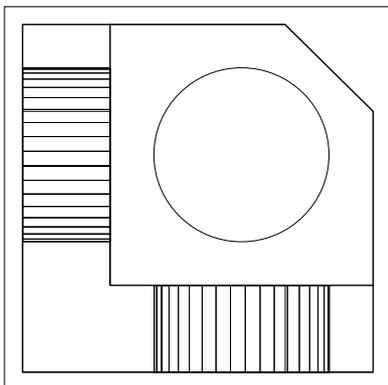
The **PLAN** command lets you restore the view of drawing from a 3D viewpoint to a 2D plan view. The axis tripod will not be displayed in this view. See also **VPOINT** command.

**Procedure:**

@CMD: **PLAN** (*return*)  
(*Drawing regenerated on X/Y projecting plane.*)

**Example:**

3-D view and Plan View of the same drawing:



# PLINE

## Draw a Polyline Entity Command

---

### Purpose:

The **PLINE** command is used to draw a polyline.

### Description:

The **PLINE** command lets you draw polylines. A polyline is a composite entity with line and arc segments joined together. So, to draw a polyline is to draw successive lines and arcs.

### The Three Drawing Modes

The **PLINE** command provides three drawing modes for you to create its forming segments successively, namely

- **Line mode** - To draw a line segment. The **LINE** command is invoked to handle the line drawing with additional **PLINE** sub-command options. This is the initial default mode when you issue the **PLINE** command, and is activated by entering the sub-command option "L" from the other two drawing modes. See **LINE** command for details.
- **Arc mode** - To draw an arc segment. The **ARC** command is invoked to handle the arc drawing with additional **PLINE** sub-command options. This mode is activated by entering the sub-command option "A" from the other two drawing modes. See **ARC** command for details.
- **Spline arc mode** - To draw a spline arc segment. A spline arc segment consists of two arc segments joined smoothly at their common tangent point. The **SARC** command is invoked to handle the drawing of a spline arc, with additional **PLINE** sub-command options. This mode is activated by entering the sub-command option "S" from the other two drawing modes. See **SARC** command for details.

Each segment of a polyline is drawn in one of these modes in the same way as that for a single primary entity. However, except for the first segment, each polyline segment receives its start point specification from the preceding segment, as they are drawn one after another.

### Closing A Polyline

The polyline being drawn can be closed to its start point automatically, if you enter the sub-command option "C" in Line mode, or "CL" in Arc or Spline Arc mode. **TwinCAD** will close the polyline by adding a closing segment that meets the start point. The closing segment may be a line, arc or spline arc, depending on the current drawing mode you are in, as described below:

- **Line mode** - A line segment is drawn directly to the start point of the polyline.
- **Arc mode** - An arc segment is drawn to the start point of the polyline. The way the last arc segment is connected to the start point will depend on which drawing mode it is in. Before the last segment is drawn, if it is in tangent mode (which is the default setting), then the closing arc segment will connect to the start point without being tangent to the starting segment. On the other hand, if it is not in tangent mode (i.e., in Free direction setting), then the closing arc segment will be made tangent to the starting segment of the polyline at its start point.

- **Spline arc Mode** - A spline arc segment is drawn tangent to the starting segment of the polyline at its start point. When it is set in Free direction mode instead of tangent mode, the closing segment will be drawn as a single arc segment (as against a double-segment spline arc), and will be tangent to the starting segment.

### Continue an Open Polyline

You may continue drawing an existing open polyline by snapping at either of its end point, via **ENDp** snap function, at the start of the **PLINE** command. **TwinCAD** will then make successive drawing of polyline segments as if they were drawn continuously from the very beginning. The **"C"** option will close the polyline properly. So you can quit the drawing of a polyline anytime, and pick it up again later.

### Undo Last Segment

You may undo the last drawn segment by entering the sub-command option **"U"**.

### Exit PLINE Command

To exit the **PLINE** command, type a null return or a **<Ctrl/C>** for the Line mode, and type **Ctrl/C** for the Arc and Sarc modes.

### Procedure:

To draw a polyline, follow the example procedure as below:

```
@CMD: PLINE (return)
(default in Line mode)
Arc/Sarc/<Startpoint of line>: (point)
Arc/Sarc/Close/Undo/Free/Deflect/<Endpoint/length>: (point)
...
( switch to Arc mode )
Arc/Sarc/Close/Undo/Free/Deflect/<Endpoint/length>: A
Radius/Direction/Free/Deflect/Close/Line/Sarc/Undo/<Endpoint>: (point)
...
( switch to Spline Arc mode )
Radius/Direction/Free/Deflect/Close/Line/Sarc/Undo/<Endpoint>: S
Line/Arc/Deflect/Free/<End point of Spline Arc>: (point)
Free/Complement/Standard/<Control point>: (point)
...
( switch to Line mode )
Line/Arc/Deflect/Free/<End point of Spline Arc>: L
( Close the polyline )
Arc/Sarc/Close/Undo/Free/Deflect/<Endpoint/length>: C
```

### Example:

# PLOT

## Plot Selected Objects or View Command

---

### Purpose:

The **PLOT** command is used to plot out selected objects from the drawing or a window view of the drawing to the output device.

### Description:

The **PLOT** command allows you to plot out either drawing entities or drawing views. The entities could be all the objects (by default) or some selected objects from the drawing; and the plotted views could be current window view or a selected window view of the drawing. The plotting device configuration is also provided directly by the **PLOT** command.

The **PLOT** command starts with the prompt:

Configuration/Entities/View/<All drawing>:

Each option item is described as below:

- C**     **Configuration**, request to configure the plotter device setup data. A GUI dialogue window will be popped up for configuring the plotting device. See later section.
- E**     **Entities**, request to select objects to plot. You will be asked to select the objects to plot.
- V**     **View**, request to plot out the drawing by window view. You will be asked to select the view of the drawing to plot.

**Return**   **Select all objects by default**, request to plot out all the drawing.

You may choose to plot out all the drawing by answering to the prompt with a null return. If you want to plot out only some selected objects, enter the sub-command option "**E**" and select the desired objects. In the case when you want to plot out a window view of the drawing, enter the sub-command option "**V**" and set up the window view to plot out. After you finish selecting the objects or setting up the window view to plot out, **TwinCAD** will determine the drawing extent of the plotting to be used in later operation.

### Plotting by Objects

Plotting by objects has its special applications in industry. For example, you are unlikely to use the cutting plotter to cut out a view of a drawing clipped by a window rather than a specific object with well-defined shape. A well-defined shape is formed by an object or a group of objects without being clipped.

The plotting functionality provided by **TwinCAD** can be object-specific. With the open architecture of the **PDF (Plotter Driver File)**, the **PLOT** command may serve as a simple and handy solution for many industry applications, such as that for cutting plotters.

The default object selected in the **PLOT** command are all the objects from the drawing. This is realized by pressing space bar to the first prompt of the command. To plot out only specific objects, you have to enter the sub-command option "**E**", and select desired objects via the Object Selection Operation.

## Plotting by View

**TwinCAD** also provides the plotting functionality based on a window-clipped drawing view. For most plotter output, a properly clipped drawing view is usually desired. The drawing view may contain the whole drawing, or only a part of the drawing.

In response to the sub-command entry "**V**", **TwinCAD** will prompt

What to plot -- Display, Extents, Limits or Window <D>:

for you to set up the view window of the drawing to plot out. The possible options are described as below:

- D**     **Display**, specifying the current view of the drawing in the display to be plotted. This is the default and will be taken if you press the space bar and return nothing.
- E**     **Extents**, specifying the view of the drawing to its extent to be plotted. This is virtually the same as to plot out all objects of the drawing.
- L**     **Limits**, specifying the view of the drawing generated by **ZOOM LIMIT** to be plotted.
- W**     **Window**, specifying the view window of the drawing defined by two points to be plotted. You will be asked to specify two points for the view window corners.

## Read the Plotting Configuration Setup

After you have determined what to plot, **TwinCAD** will read the current plotting configuration saved in disk and load the Plotter Driver File as specified in the configuration. If the configuration file can not be found, **TwinCAD** will give the error message:

Can't find plotter setup data, requiring configuration.

and display the configuration dialogue window for you to complete the configuration first. If the Plotter Driver File specified in the configuration file can not be found from the disk, **TwinCAD** will give the error message:

Can't find PDF file: xxxxxxx

and enter the configuration dialogue window. If everything is OK, **TwinCAD** will be able to read the default plottable area of the plotter from the **PDF** file and report it as below:

Maximum plottable area reported from PDF: *nnn.w* x *nnn.h* mm

## Set Plotting Area on Plotter

The plotting operation requires you to establish the correlation between the extent of the drawing image to plot out and the plotting area on the plotter by specifying the plotting origin at the plotter and the scale factor for the drawing.

To help you set up this correlation, **TwinCAD** will clear the drawing window, set up a proper size of view, and draw out:

- **a white rectangle with solid lines**, representing the selected paper (always placed at the lower left of the plotter) from the configuration setup,
- **a white rectangle with dashed lines**, representing the plottable area of the plotter (the plotter table size) reported by the selected **PDF**,
- possibly **some red rectangles with solid lines**, representing previous plotted areas, assuming that you are plotting drawings or views on the same paper repeatedly, and
- **a yellow rectangle with solid lines**, representing the current plotting area determined by the drawing extent under current active setting of the scale factor and plotting origin.

Note that the actual color of these rectangles will depend on the current pen color assignments as well as the video color palette assignments in the system initial file.

From the plot assistance as described above, you can see exactly where the drawing will be plotted on the paper, and place the drawing on the paper much easier.

**TwinCAD** will create another rectangle of the same size as the one representing the current size of the plotting image for you to drag around in the window, and prompt:

```
Set plot area -- Clear/Fit/Preview/Quick-view/Rotate(nn°)/Same-mapping/  
/Last-area/<scale(nnn)/origin(xxx,yyy)>:
```

You may change the plotting origin (the lower left corner of the current plotting area) by designating a point to the prompt, or change the size and the orientation of the plotting image by the other options provided with the prompt. You may directly observe the display area of the drawing on the paper from the screen and adjust it to your satisfaction. **TwinCAD** will prompt again and again, until you press the space bar to finalize the setting of current plotting area and to start the plotting.

The following paragraphs describe these options in details.

### Set the Plotting Area with a Scale factor

You may change the size of the plotting image by entering a different value of scale factor to the prompt. The size of the dragged rectangle will be changed accordingly, and so is the current plotting area represented by the yellow rectangle.

### Set the Plotting Area by Fitting

You may directly set up the size and location of the plotting area on the plotter and fit the plotting image into it, by entering the sub-command option "**F**". You will be asked to specify two points which define a rectangular area to fit the drawing. The fitting will determine the effective scale factor and the plotting origin.

### Set the Plotting Area by Mapping with the Last Plot

You may specify to use the same mapping relationship as that for the last plotting activity by entering the sub-command option "**S**", which means the Same drawing to plot. This option is useful in plotting the missing part of the same drawing (such as the drawing on a frozen layer).

### Set the Plotting Area by the Last Plotting Area

You may specify to fit the drawing to the plotting area specified by the last plotting activity, by entering the sub-command option "**L**". This is also the default setup when the **PLOT** command is entered.

Note that there is a major change after version 2.0 in this default setup. The version before 2.0 takes the "**S**" option as its default setup; while the version after 2.0 takes the "**L**" option instead.

### Rotation of the Plotting Image

The image of the drawing to plot out may be rotated with an arbitrary angle. Entering the sub-command option "**R**" to the prompt, you may specify the value of this rotation angle in unit of degree, and **TwinCAD** will prompt:

```
Enter Drawing Rotation Angle (nnn°):
```

If the plotting image is based on object selection, the drawing extent of those objects to plot out will be recalculated under that specified angle of rotation, and the dragged rectangle will be recalculated by the current scale factor. However, if the plotting image is taken from a view of the drawing, the extents of the view window will be rotated after the calculation.

### Clear the Memory of Plotted Areas

You may enter the sub-command option "**C**" to clear all the red rectangles on the screen. **TwinCAD** will memorize up to 32 such plotted areas on a paper, until you clear them out. These red rectangles are helpful when several drawings are to be plotted on the same paper.

### Preview of the Plotting Image

If the yellow rectangle can not satisfy you in visualizing the final plotting result, you may enter the sub-command option "**P**" for a preview of the plotting job. **TwinCAD** will actually process the plotting activity without sending the codes to the plotter. So, from the preview, you may see in advance how the drawing will be plotted, and where it will be plotted.

If your drawing is very complicated, the preview may be a little bit slow just as it will actually go through in the plotting activity. You may choose to enter the sub-command option "**Q**" for quick preview, if you are only concerned about the placement of the drawing image. The Quick-view will generate the drawing in a much faster way by

1. Ignoring the pen sequences (so it scans the drawing only once),
2. Turning off the solid color fill, and
3. Inhibiting the generation of Region Hatch.

### Start the Plotting

To start the plotting, you must press the space bar or return key to finalize the current setting of the plotting area (as shown by the yellow rectangle). After that, **TwinCAD** will be ready to process the plotting activity.

### Generating the Plot File

At this point, you may choose to produce a plot file by redirecting the output of the plotting data to a specific disk file, rather than sending them directly to the plotter through the hardware channel, by answering to the last prompt:

Plotting to a disk file ? <N>:

If your answer is "Y", **TwinCAD** will pop up the file window for you to specify the name of the plot file. The file extension of this plot file is always "**PLT**".

### Plotting in Progress

At last, **TwinCAD** prompts the messages:

Plotting to device: xxxx

Plotting activity in process, type <Ctrl/C> to stop...

and starts plotting. During the plotting, you can observe the whole process from the screen, where a counterpart of the plotting is also drawn (as the preview). You may type <Ctrl/C> to abort the plotting activity, when necessary.

After the plotting, **TwinCAD** will make a short beep and report:

Plotting over, total *nnnnn* vectors plotted. Press any key:  
and wait for you to press any key to end the command.

## Configuration of Plotter Setup Data

You may configure the plotting device and other related plotting parameters by entering the sub-command option "C" to the first prompt when the **PLOT** command is entered. **TwinCAD** will pop up a dialogue window for you to configure the current plotter setup data. This configuration dialogue window is also popped up automatically when

- **TwinCAD** can not find the plotter setup data from the known disk path, or
- **TwinCAD** can not load in the **Plotter Driver File** from the current configuration of plotter setup data.

## The Plotter Setup Data

The plotter setup data includes the following items:

- **The PDF filename** - Specifying where to load the **Plotter Driver File**.
- **I/O Connection** - Specifying the hardware channel the plotter device is connected to. This includes the communication parameters setup if the RS-232c interface is used.
- **Plotting Pen Setup** - Establishing the relationship between drawing colors and the real pen numbers used in the plotter. This includes the setup of the pen width, plotting speed and line type for that pen.
- **Paper Setup** - Specifying the size of the paper to use on the plotter. This paper size is set up only for reference.
- **Plotter Calibration Setup** - Specifying the calibration factors for the plotter.

You set up these data by picking at the corresponding items from the dialogue window. For example, to set up the I/O connection, pick the field labeled with '**I/O connection**', where the current setup is displayed, and **TwinCAD** will pop up a selection window for you to select one supported I/O channel from within it.

**TwinCAD** will save the plotter setup data automatically to current logged disk with the name "PLOTTER.SET", when you finish the setup operation by picking the **[OK]** button from the window.

## Select Plotter Driver File (PDF)

The field labeled with '**PDF filename**' from the configuration window contains the name of the selected **Plotter Driver File** for the current plotter setup. You pick up the field, and **TwinCAD** will pop up a file window for you to select from disk the proper **PDF** file.

The **Plotter Driver File** or **PDF** is a file that controls the output conversion formats for all plotting commands. It is used to translate the plotting commands issued by **TwinCAD** into the equivalent plotter commands that can generate the drawing on the plotter. You have to select one such file that has been well-defined for your plotter output. See appendix for detailed description to the **PDF** file content.

## Select I/O Connection

I/O connection means the hardware channel through which the plotting data are sent. You may select any one from **LPT1:** to **LPT4:** for printer port output or from **COM1:** to **COM4:** for RS-232c communication port output. If an RS-232c communication port is selected, you will

have to set up the proper communication parameters, such as baud rate, word length, parity and number of stop bits, for that port. Additional dialogue window will be popped up for you to set up these parameters. See your plotter's operation guide for proper setup value of these parameters.

## Set up Plotting Pens

**TwinCAD** lists out a table of 16 colors in a slide sub-window in the dialogue window. Each color has a field of pen number, pen width, plotting speed and line type, as described below:

- **Pen number** - Specifying which pen to use for that color. If you have several pens on the plotter, you may need to tell **TwinCAD** which pen to use for each color in the drawing. You may change the pen numbers each time you change the pens on the plotter.
- **Pen width** - Specifying the line width of the plotting pen in units of millimeters (mm). This value is used in solid color fill. Be sure to enter the correct value. To some laser printer that emulates a plotter, this value is also used to set the width of the line used for the pen number.
- **Plotting speed** - Specifying the speed number used for the pen. Enter a **0** for default speed (usually the maximum speed). This is a plotter dependent parameter. The unit of this pen speed is dependent on the plotter, but is usually in cm/sec.
- **Line type** - Specifying the line type number used for the pen. Unless you have special reason, you should enter a **0** for default solid line type. This is a plotter dependent parameter.

To change the plotting pen parameter settings of a color, select the color and **TwinCAD** will let you modify these fields in sequence.

## Set up Paper Size

The paper size setting tells **TwinCAD** to draw a white rectangle in the determination of the plotting area for reference purpose. You may set the size of the paper directly by entering the width and height value to the fields of "**X**:" and "**Y**:". Or, you may prefer to select the paper of given size from the field of "**Paper**:".

When you pick up the field of "**Paper**:", a slide window for selecting papers of different sizes will be popped up. Simply select the one you need. A file named "**PLOTTER.SIZ**" contains the definition of these papers. You may add new paper sizes which you commonly use to the file.

**TwinCAD** ships the file "**PLOTTER.SIZ**" with the following default content:

```
"A0 " , 1189 , 840
"A1 " , 840 , 586
"A2 " , 586 , 420
"A3 " , 420 , 297
"A4 " , 297 , 210
"A5 " , 210 , 148
"B5 " , 257 , 182
"A3-P " , 297 , 420
"A4-P " , 210 , 297
"A5-P " , 148 , 210
"B5-P " , 182 , 257
"ANSI-A " , 268 , 203
"ANSI-B " , 406 , 254
"ANSI-C " , 533 , 406
"ANSI-D " , 838 , 533
```

```
"ANSI-E" , 1092 , 838
```

```
...
```

The first quoted string is the name of the paper, followed by its width and height in unit of millimeter.

## Calibration of Plotter

You may assume that the plotter you use meets the manufacturer's specification for the accuracy of scale. In such case, the calibration factors for the plotter will all be **1**, which are also the default. However, if it does not meet the specification, you may correct the possible mis-scaled plotting by entering the correct calibration factors.

To obtain the correct calibration factors, follow the procedure below:

1. Draw a square box of fixed size **S**, say 100 mm.
2. Plot it out to the plotter in 1:1 scale.
3. Measure the actual size of the box from the paper. Let the result of measurement be **X** and **Y** for the width and height of the box respectively. Then the calibration factors for both axes will be:

$$C_x = S/X, \text{ and}$$

$$C_y = S/Y$$

## The Setup File: PLOTTER.SET

TwinCAD always saves the current plotter setup data to the file: "**PLOTTER.SET**" in current logged-in disk and directory, although it may read the file from the path specified by TCADPATH if a file is missing from the logged-in directory.

## Special Note

A system variable "**PPLSYMBOL**" is used to specify the paper layer symbol for this **PLOT** command. If this symbol is present and loaded, then the paper size display (white rectangle) will include a full size display of this symbol. It will serve as a pre-printed paper used for the plotting job. Note that this symbol is drawn on the screen only for visual reference and will not be plotted out anyway.

Two system variables "**MAXPVLEN**" and "**MAXPVANG**" are used to control the segmentation of curve to the plotter. The value **MAXPVLEN** specifies the maximum length of the line segment used, and **MAXPVANG** specifies the maximum interval of the curve (arc) angle allowed to be approximated by a line segment. Curve segmentation by line segments are automatically done when

- The curve is not an arc (in the plotting paper space), or
- The X and Y plotter calibration factors are not equal, or
- The center or radius of the arc is outside of the allowable coordinate ranges (in plotter unit specified by the PDF file).

## Procedure:

- **To configure the plotter setup data, follow the procedure below:**  
 @CMD: **PLOT** (*return*)  
 Configuration/Entities/View/<All drawing>: **C** (*return*)

*(TwinCAD pops up dialogue window)*

Configuration/Entities/View/<All drawing>:

- **To plot out selected objects, follow the procedure below:**

@CMD: **PLOT** *(return)*

Configuration/Entities/View/<All drawing>: **E** *(return)*

Select Objects (+): *(do so)*

Maximum plottable area reported from PDF: *nnn.w x nnn.h mm*

Set plot area -- Clear/Fit/Preview/Quick-view/Rotate(*nn°*)/Same-mapping/

/Last-area/<scale(*nnn*)/origin(*xxx,yyy*)>: *(point)*

Plotting to a disk file ? <N>: **N** *(return)*

Plotting to device: *xxxx*

Plotting activity in process, type <Ctrl/C> to stop...

*(Plotting in process)*

Plotting over, total *nnnnn* vectors plotted. Press any key:

- **To plot out a selected view, follow the procedure below:**

@CMD: **PLOT** *(return)*

Configuration/Entities/View/<All drawing>: **V** *(return)*

What to plot -- Display, Extents, Limits or Window <D>:

...

### **Example:**

# 'PLOTOPT

## PLOT/PRPLOT related Options Setup Command (TCL)

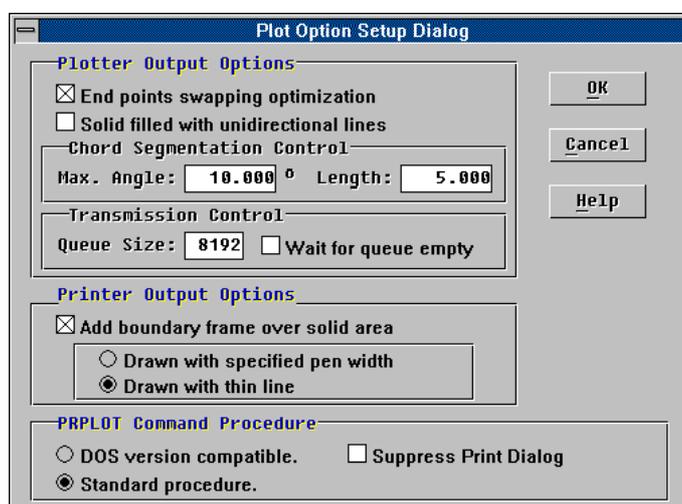
### Purpose:

The **PLOTOPT** command is used to setup drawing plot-out related options via a GUI dialogue operation.

### Description:

The **PLOTOPT** command lets you access the drawing plot-out related options via a GUI dialogue window operation. Some of these options are controlled by specific system variables.

**PLOTOPT** will pop up the dialogue window as shown below:



The following describes these screen items in details.

### Plotter Output Options

There are a few options in this group.

#### End point swapping optimization

If this option is enabled, **TwinCAD** will check for every output vector for nearest end point swapping optimization. That is, if a plotting vector should be output to move the pen from A to B, while the position B is nearer to the current pen position, it will be output to move the pen from B to A. This optimization will reduce the total length of pen movement and thus will reduce the total time spent in pen movement.

However, if the output device is not a real pen plotter, this optimization will not bring you any real benefit. You may like to turn it off for faster output processing. Likewise, if your application requires the output pen stroking sequence to follow a predefined way, you will have to turn this option off.

### Solid filled with unidirectional lines

If this option is enabled, and an area must be filled with scan lines for the solid color, then **TwinCAD** will fill it with unidirectional scan lines, instead of swinging the scan lines bi-directionally to save the pen-up and pen-down operation. This option is suitable for non-pen plotter output.

### Chord Segmentation Control

If the plotter does not support circular arc plotting command (as indicated by the loaded PDF file), **TwinCAD** will replace these circular arcs with successive chord lines. You may control the fineness of this circular arc segmentation by specifying

**Max. Angle**            The maximum chord angle allowed.

**Length**                The maximum length of the chord.

### Transmission Control

**TwinCAD** conducts the parallel/serial port communication via Windows API calls, which manipulate a queue buffer for the data transmission. If you are using the serial communication port for the plotting device, you may specify the size of this transmission queue. The maximum size of this queue is 65535 bytes. The internal default is 8192 bytes.

Whenever a plotting command code is generated and is to send through the communication port, it will be stored first into the transmission queue for Windows to transmit it through the line transparently. After all codes are generated to the transmission buffer, **TwinCAD** will be free from waiting and be able to continue its CAD service, while Windows does the transmission in the background. So, the larger the buffer, the sooner **TwinCAD** will end the plotting processing.

However, if you are going to plot several drawings in a batch processing manner via the use of a TCL application program or pre-written command script, or there are chances that other Windows applications may also use the same communication port for the same purpose, you should make a check on the option: "**Wait for queue empty**", such that **TwinCAD** will continue locking the communication port for transmission until the transmission queue is empty. This will prevent the corruption of the transmission queue after **TwinCAD** has logged off the channel.

### PRPLOT Command Procedure

You may issue **PRPLOT** command to print out the current drawing to the printer. However, there are two different operating procedures:

**DOS version compatible**    If you are familiar with the **TwinCAD**'s DOS version, or you have a TCL application that relies on the compatibility with the DOS version, you may choose to process the **PRPLOT** command in the DOS version compatible way (which pop-ups a dialog window first).

**Standard procedure**    The standard **PRPLOT** command procedure will let you setup the drawing frame on the paper first, where you may enter options to configure the printer, paper size and other things. It is more compliant with the Windows convention in printing documents.

If the **PRPLOT** command procedure is not driven by a command script, **TwinCAD** will call Windows Common Dialog to access the printer, where you may have the last chance to

change settings or cancel the printing. You may skip this dialog by making a check on the option: "**Suppress Print Dialog**".

### Other Buttons

After completing your option setup, you have to press the **[OK]** button to confirm the change and terminate **PLOTOPT**. Or, you may press the **[Cancel]** to quit **PLOTOPT** operation. You may also press <ESC> key or the right mouse button to quit the operation.

### Special Notes:

The **PLOTOPT** command is an external command provided by the TCL program file "PLOTOPT.TCL" or "PLOTOPT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **PLOTOPT** command, you may solve the problem by copying the "PLOTOPT.TCL" or "PLOTOPT.TCA" to the COMMANDS sub-directory.

### Procedure:

Enter the **PLOTOPT** command to the system command prompt:

```
@CMD: PLOTOPT
```

```
...[Dialogue Operation]...
```

# POINT

## Draw a Point Entity Command

---

### Purpose:

The **POINT** command is used to create a point entity or set the display mode and size of the point entities.

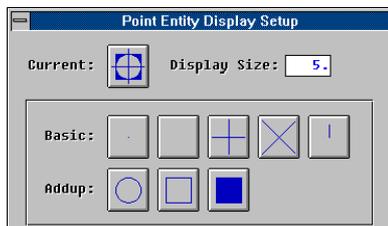
### Description:

The **POINT** command lets you create a point entity or set the display mode and size of the point entities. When you enter the **POINT** command, **TwinCAD** will prompt:

Display/<define point>:

asking you to designate the position of the point to create. After you have designated the point, the point entity is created and the command ends. The Last Point coordinate is also updated by the designated point.

The sub-command option "**D**" is used to set up the display control of the point entities. There are many ways to display a point entity, so that you can identify it easily. When you enter the sub-command option "**D**", **TwinCAD** will pop up a Point Entity Display Setup window, as shown below, for you to set up this display control.



The display control includes two parts:

- **Display Type** -- The type of a point entity display. This defines the shape of the marker for the point entities. It consists of a basic shape, such as a dot, X-cross, "+" cross or nothing, with possible add-up shapes, like a square, circle with or without solid fill. See the figure below.
- **Display Size** -- The size of the marker to display. If it is positive, the size is in drawing unit. If it is negative, the size will be in the unit of percentage to the size of the current drawing window.

You simply pick up the screen button to select the marker you desire. The current shape of the point marker is also shown in the display, so that you will know what you have done.

The display control of the point entities is global. The display of all point entities will be changed to the new setting after the regeneration of the drawing.

### Procedure:

@CMD: **POINT** (*return*)

Display/<define point>:

...

### Example:

# POLYGON

## Draw Regular Polygon Command

---

### Purpose:

The **POLYGON** command is used to draw a regular polygon with any number of sides from 3 to 1024.

### Description:

The **POLYGON** command lets you draw a regular polygon with a given number of sides from 3 to 1024. The size of the polygon may be specified by the radius of a circle which is inscribed or circumscribed to the polygon, or by the length of an edge. The resulting polygon will be a closed polyline comprising of line segments.

When you issue the **POLYGON** command, you will be asked to specify the number of sides of the polygon to draw, as in the prompt:

Number of sides (*n*):

where *n* is the number of sides from the last created polygon. After that, you may select the method to draw the polygon by answering to the prompt:

Edge/Inscribing-circle/Circumscribing-circle/<Center of polygon>:

### Draw Polygon by a Center Point and a Starting Vertex

You may draw the polygon by designating its center point to the prompt. **TwinCAD** will then prompt

Starting vertex:

asking you to designate a point as the starting vertex of the polygon, which determines the size and the orientation of the polygon. On specifying the starting vertex, you may observe the formation of the polygon from the dragging image as you move the cursor pointer.

### Draw Polygon by Specifying a Polygon Edge

You may draw the polygon with a given polygon edge, by entering the sub-command option "E" to the prompt. **TwinCAD** will display a prompt similar to the **LINE** command for you to specify the first polygon edge:

Polygon starting edge, from point:

Second point of starting edge:

The polygon will be drawn from the starting edge, followed by a series of copied edges rotated in counter-clockwise direction. You may observe the formation of the polygon, when specifying the second point of the edge, from the dragging image as you move the cursor pointer.

### Draw Polygon by Specifying a Circumscribing Circle

You may draw the polygon within a circle which circumscribes to it, by entering the sub-command option "C" to the prompt. **TwinCAD** will display a prompt similar to the **CIRCLE** command for you to specify the circle:

Circumscribing Circle -- 3P/2P/<Center point>:

The polygon is then determined by the circumscribing circle with the starting vertex at zero degree or at a specific direction relative to the center, depending on how the circle is determined. You may observe the formation of the polygon, when the dragging circle appears, from the dragging image as you move the cursor pointer.

### Draw Polygon by Specifying an Inscribing Circle

You may draw the polygon based on a given circle which inscribes to it, by entering the sub-command option "I" to the prompt. **TwinCAD** will display a prompt similar to the **CIRCLE** command for you to specify the circle:

Inscribing Circle -- 3P/2P/<Center point>:

The polygon is then determined by the inscribing circle with the inscribing point on the starting edge at zero degree or at a specific direction relative to the center, depending on how the circle is determined. You may observe the formation of the polygon, when the dragging circle appears, from the dragging image as you move the cursor pointer.

### Draw Polygon on PUCS-Plane

The **POLYGON** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection, and allows you to create polygons on the projected plane from the virtual space directly.

#### Procedure:

- **To draw a hexagon with a given inscribed circle:**

@CMD: **POLYGON** (return)

Number of sides: **6** (return)

Edge/Inscribing-circle/Circumscribing-circle/<Center of polygon>: **I** (return)

Inscribing Circle -- 3P/2P/<Center point>: (Define the circle, see **CIRCLE** command)

...

A hexagon is drawn such that the given circle is inscribed to it and the first point of inscription starts from zero degree.

- **To draw a pentagon with a given circumscribed circle:**

@CMD: **POLYGON** (return)

Number of sides: **5** (return)

Edge/Inscribing-circle/Circumscribing-circle/<Center of polygon>: **C** (return)

Circumscribing Circle -- 3P/2P/<Center point>: (Define the circle, see **CIRCLE** command)

...

A pentagon is drawn such that the given circle is circumscribed to it and the first vertex starts from zero degree.

- **To draw an octagon with a given center and starting vertex:**

@CMD: **POLYGON** (return)

Number of sides: **8** (return)

Edge/Inscribing-circle/Circumscribing-circle/<Center of polygon>: (point)

Starting vertex: (point)

An octagon is drawn such that one of its vertices and its center are located at the given points.

- **To draw a decagon with a given edge:**

@CMD: **POLYGON** (return)

Number of sides: **10** (return)

Edge/Inscribing-circle/Circumscribing-circle/<Center of polygon>: **E**(return)

Polygon starting edge, from point: (point)

Second point of starting edge: (*point*)  
A decagon is drawn such that one of its edge is as specified.

# PRPLOT

## Graphic Display List Print/Plot Out Command

---

### Purpose:

The **PRPLOT** command is used to print/plot out the drawing image on the screen, from the graphic display list, to the output device. It produces a hard copy of the drawing image.

### Description:

The **PRPLOT** command lets you generate a hard copy of the drawing image by converting its display lists from the graphic device to the printer/plotter. The output device may be a dot matrix printer, laser printer, ink-jet printer or a plotter, or a disk file in a specific image file format, depending on the device driver selected to use.

Since it produces a hard copy of the drawing image from the screen, you will be asked at first to set up a view window to output by the following prompt:

What to plot -- Display, Extents, Limits or Window <D>:

**TwinCAD** will take the current display as the default. You set up the view window to output by entering one of the sub-command options as below:

- **D - Display**, specifying the current view window to output. This is also the default when you answer with a null return.
- **E - Extents**, specifying the view window to the drawing extents.
- **L - Limits**, specifying the view window to the drawing limits.
- **W - Window**, specifying the view window by designating two points in diagonal direction. You will be asked to specify the new window as if you were issuing the ZW command. Only the drawing within the window defined by the two points will be scaled to output.

After the view window is set up, **TwinCAD** will regenerate the drawing within the view window to obtain the fresh display lists to output. Note that the generation of this fresh display lists may look distorted on the screen. This is mainly because the aspect ratio of the output is recalculated for the output device, and not for the drawing window.

And then, **TwinCAD** will load in the device driver overlay currently set up to handle the display lists. If this overlay file does not exist, or there is no such setup, a device driver selection window will be automatically popped up for you to select the proper one to load.

### Device Setup Dialogue Window

Once the driver overlay is loaded, **TwinCAD** will pop up a device setup dialogue window for you to set up the parameters that control the image output. The parameters can be classified as below:

- **Device Type** - specifying which type of output device to use, you can either choose from current-supported drivers or request to load another driver.
- **Connection** - specifying either output channel (**LPTn:** or **COMn:**) for the real device or the filename for disk image file output.
- **Resolution** - specifying output resolution from possible alternatives.

- **Plotout Area** - specifying the area where the display image is placed in the output device.
- **Orientation** - specifying whether the output image will be rotated or not.
- **Pen Width and Solid Pattern Control** - specifying the width of each color pen and also the patterns used for each solid color fill in the image output.
- **Adding Frame** - specifying whether to add a frame around the plotout area.
- **ORing Pattern Fill** - specifying whether to generate the pattern fill by ORing operation, instead of PUTting operation.
- **Direct Port I/O** - specifying whether to send the graphic data directly via port I/O to the printer, or via the BIOS I/O call.

Once these parameters are set up, they are saved into a disk file. Therefore, you won't need to specify them each time you enter **PRPLOT** command. From time to time, however, you may want to change the pen width and its pattern control, or the plotout area and the filename for disk image file output. Occasionally, you may also want to switch the device driver to the one that produces a specific disk image file or the one that handles a specific printer.

The in-depth discussion on parameter setup is described in the following sections.

## Select the Device Type

**TwinCAD** supports two basic types of driver overlays that handle the generation of the graphic image. One is for the real devices, like dot-matrix printers, laser printers, plotters and so on. The other is for disk image file output in a specific format like PCX, TIFF, GEM/IMG, BMP, etc.

Each driver overlay may handle more than one kind of device or disk format. For example, a driver overlay for Epson printers supports Epson 9-pin as well as 24-pin printers. You may choose either kind from the selection list of the **Device Type**, if the said driver overlay is loaded. Another example is the driver overlay that produces disk image file in PCX format. You may choose to produce the image in B&W format, reversed or not, or the 16 color format, etc., from the selection list of this **Device Type**. The selection list will be popped up when you pick up the field of this Device Type.

In addition to the possible devices or disk formats that the driver overlay may support, there is also a special item in the selection list of the **Device Type**:

<Others from Different Driver>

which is for you to change the device driver overlay. Pick up this item, and **TwinCAD** will search the disk for all the device driver overlays and pop up a slide window, containing the functional name of them, for you to select the desired one. Once it is selected, it will be loaded and become the current active driver overlay.

## Setting Output Connection

The output connection means the hardware channel through which the image data are transmitted, or the destination where the image data are stored, depending on the type of the driver overlay currently loaded.

If the driver overlay is for real devices, then when you pick up this field, you will have to select the output channel from a pop-up window containing the possible selections from **LPT1:** to **LPT4:**, and **COM1:** to **COM4:**. After a communication port is selected, a parameter setup window will also be popped up for further setup, such as baud rate, parity, word length and so forth.

If the driver overlay is for disk image output, then when you pick up this field, the file window will be popped up for you to specify the output filename.

Note that the I/O parameter setups for the real devices will not be changed by the switching of device drivers, unless you change them explicitly.

## Setting Output Resolutions

If the driver overlay is for real device, the possible output resolutions supported will depend on the device. Pick up this resolution field, and **TwinCAD** will pop up a selection window containing all the supported resolution alternatives from the device. You simply choose the one desired.

If the driver overlay is for disk image file output, the output resolutions may be specified to any reasonable values you desire. So, you will have to enter the values by yourself. Usually, the resolution of 300 x 300 dpi is recommended. Note that the aspect ratio of the image file is controlled by this resolution setting.

The resolutions are always in unit of **DPI** (dot per inch).

## Setting the Plotout Area

Assuming that there is a paper with the origin at its top left corner, you have to decide where to put the plotout image onto the paper and how large the image is desired. The size of the image is defined by its height and width. The position of the image with respect to the paper is determined by the top margin (space on the top) and left margin (space on the left) specified. These are specified in unit of inch or centimeter (you can pick on the unit display and toggle the units).

The output image can be scaled so that the width of the image will meet the specification as required. As the aspect ratio of the drawing image must be maintained in the output, the output image will be padded with white spaces or chopped off in height, if the aspect ratio of this plotout area is not the same as the drawing image. However, you may specify the output image height to be floating with the drawing image by setting it to **0**, so that the actual output image height will be calculated according to the actual aspect ratio of the drawing image.

## Using the [SetFrame] Option

Before you can preview the plot image on the screen, you will have to set up the related parameters such as top margin, left margin, plot width and plot height first. Since it is a cumbersome work for you to do all the setting, **TwinCAD** provides a "**SetFrame**" utility for you to work out the parameter setting much easier.

To start the operation, you may pick up the button labeled as "**SetFrame**", then **TwinCAD** will clear up the drawing area, set up a paper ruler like one in the desktop publishing software, and let you set up the image frame interactively. The image frame thus set up will define the four parameters for the plotout area.

### TwinCAD prompts

```
Set Print area -- Fit/Normal/Rotated/<scale(nnnn)/origin(nnn,nnn)>:
```

while letting you drag the rectangle in the output image size.

The operation principle is about the same as that for **PLOT** command. A yellow rectangle representing the current setting of the plotout area is drawn. You can clearly see where it is located on the paper and how large it is. The initial scale factor is calculated based on the extent of the drawing image and the current setting of the plotout area.

The coordinates at origin will be defined as the left margin and the negative value of the top margin, in units of millimeter.

The possible entries to this prompt are described below:

- **F** - Fit, specifying the image frame by two corners.
- **N** - Normal, specifying the image output to be in the normal orientation. The image frame will be recalculated based on the current setting of scale factor.
- **R** - Rotated, specifying the image output to be rotated by 90°. The image frame will be recalculated based on the current setting of scale factor.
- **(value)** - Scale value, specifying the required scale of the drawing image to output. The image frame will be recalculated based on this scale value.
- **(point)** - Point, specifying the origin point of the image frame. Note that the origin of the image frame is located at its upper left corner. Once this origin is specified, the image frame will be taken as the plotout area and the operation is terminated.
- **(space bar)** - space bar or null return, specifying to accept the current setting of image frame for the plotout area and terminate the operation.

The image frame returned by the **SetFrame** operation will define the aforementioned four parameters. Note that the specification of the top and left margins will be meaningless to the disk image file output.

## Set Image Orientation

There are only two possible orientations supported by **PRPLOT** command. One is the original orientation of the drawing image, which is the same as the normal orientation of the printer (usually portrait); and the other is to rotate the output image 90° clockwise (usually landscape).

## Pen Width and Solid Pattern Control

The drawing image output contains lines and solid fills of different colors. When it is output to the printer/plotter, different pens can be used for different colors. Different pens may mean different colors of ink used, or different widths of the lines produced. This makes the drawing output to the printer/plotter more expressive. But for a black and white dot-matrix or laser printer, it seems too ideal for the user to reach the same expression power as the plotters in this respect.

**TwinCAD** supports pen width and solid pattern fill capabilities for the printer/plotter image output. Each color of the lines in the output image is drawn by different pen of possibly different width. And each color of the solid fill is filled with different dot pattern. The pen width is in units of millimeter. The dot pattern is in 8 bits by 8 bits. If you set the pen width with a negative value, its corresponding pattern will be made coarser by expanding the 8x8 pattern to 16x16 pattern.

To set the output pen width for each color of pen, pick up the pen width field of the color and then enter the desired value. A zero pen width will disable the function. The pen width value will be converted and rounded to the nearest dot width unit before use. So, the maximum value and the accuracy of the pen width depend on the resolution of the device.

The driver overlays of current release support a maximum pen width of 24 dots. If you are printing in 300 DPI, this will be about 2.0 mm, which is sufficient for most of the case. If the pen width value is too large to support, then the largest one will be used instead.

One thing worth mentioning is the shape of the pen. The shape of the pen used in the output image is round, and assumes an aspect ratio of **1:1**. So, if the aspect ratio of the device output resolution is not 1:1, the actual width of the lines so generated will vary with their orientation. To minimize this effect, choose the output resolution of which the aspect ratio is 1:1 or closest to 1:1. For example, the Epson LQ-series printer supports 360x180dpi, 180x180dpi and many other resolutions. Don't use the 360x180dpi resolution if you are going to print the output image with the pen width control. Use 180x180dpi instead.

For solid color fill, the driver overlay will generate a pattern fill instead. The patterns currently used for each color can be seen from the setup window. To change the pattern for a particular color, pick up the pattern directly. **TwinCAD** will open a slide window of pattern selection for you to select the appropriate one. **TwinCAD** needs the file "**PATTERN.TBL**", which is also shipped with the package, to open the pattern selection window. To disable the pattern fill of a particular color, select the empty pattern.

### Add Frame to Plotout Area

An additional feature provided from the parameter setup window is the option to add a rectangle of frame around the plotout area. If this option is enabled (a cross mark in the item box), the rectangle will be drawn automatically in color number 15 (usually a white color).

### ORing Pattern Fill

An option to generate the pattern fill by ORing operation is provided from the parameter setup window. If this option is selected, the pattern filling will be generated by ORing with the existing pattern in the printing area. If this option is not selected, the generation of a pattern filling will overwrite the printing area.

### Direct Port I/O

An option to send the graphic data directly via port I/O to the printer is provided from the parameter setup window. If this option is selected, all the graphic data will be sent directly through printer port I/O, which will be much faster than BIOS I/O call. But, for slower device such as dot-matrix printer, using this option may cause problem. So, it is better turn off this option for dot-matrix printers.

### The Printing of the Image

After proper setup of these parameters (which will be saved to disk and recalled next time), you can start the printing by picking the [ **OK** ] button. A message window will come out to indicate that the printing activity is in process. During the printing, you may type **Ctrl/C** to stop the printing immediately.

All the example drawings in this manual are prepared with **PRPLOT** command in **GEM/IMG** format.

### Script-driven Plotting

The **PRPLOT** command has been enhanced, after V2.01, to accept commands from the menu script directly without entering the Device Setup Dialogue Window for the printing setup. This enhancement has made it possible to batch printing the drawing files by using a control script either from a script file or from a TCL application.

If the menu script is active just before **TwinCAD** popping up the dialogue window, **TwinCAD** will read the menu script first to set up the printing parameters, and optionally, print the drawing directly. The following is a list of control formats supported by **PRPLOT** to read commands from the script:

"L <i>nn</i> "	Set Left Margin, where <i>nn</i> is the setting value.
"T <i>nn</i> "	Set Top Margin, where <i>nn</i> is the setting value.
"W <i>nn</i> "	Set Printing Width, where <i>nn</i> is the setting value.
"H <i>nn</i> "	Set Printing Height, where <i>nn</i> is the setting value.
"S <i>nn</i> "	Set Printing Area by Scale Factor, where <i>nn</i> is the scale factor, assuming the drawing unit in millimeter.
"P <i>n mm</i> "	Set Pen width, where <i>n</i> is pen number, <i>mm</i> is the pen width in unit of millimeter.
"U <i>name</i> "	Use and load specific printer driver set file, where <i>name</i> is the name of the file.
"O <i>name</i> "	Set output file name if applicable, where <i>name</i> is the name of the output file.
"R "	Set Rotated Printing Orientation.
"N "	Set Normal Printing Orientation.
"I "	Set Unit in Inch. The following margin setting will be in units of inches.
"C "	Set Unit in Centimeter. The following margin setting will be in units of centimeter.
"F "	Request to fit the drawing to current Width/Height setting.
"\"	Request to switch to the GUI dialogue window.
" " or "^M"	Request to start printing using current setup.
"^C"	Set the data and quit the command, useful to setup data only.
<b>Else</b>	All else characters are ignored.

At the end of the script, if no carriage return or space bar is entered to start the printing, **TwinCAD** will enter the GUI dialogue mode with the preset data.

**Example 1.:** to output a drawing to fit in an A4 size paper, you may use the expression:

```
command("PRPLOT E I L 0.5 T 0.5 W 7.5 H 10 F ^M");
```

from within a TCL application or the menu script.

**Example 2.:** to output a drawing by two points in a specific scale factor:

```
command("PRPLOT W p1 p2 I L 0.5 T 0.5 S 2.0 ^M");
```

**Example 3.:** if the two points are to be specified by the operator:

```
command("PRPLOT W \\\ L 0.5 T 0.5 S 2.0 ^M");
```

## Redirect Output to a Print File (PRT)

An additional feature to redirect the printer output to a disk file is supported by **PRPLOT** command after V2.0. An additional Device Connection named "**FILE:**" is added to the selection list (among the **LPTn:** and **COMn:**), if the selected printer driver supports this feature.

To redirect the output to a specific file, you must choose the "**FILE:**" from the Device Connection selection list. A file selection window will then pop up for you to specify the output file name. Once the file name is entered, **TwinCAD** will proceed to print the drawing.

If the plotting to file is successfully completed, **TwinCAD** will report:

Print File: xxxxxxxxxx (size=nnnnnn bytes) is produced.

If the printer spooler (**PRINT** from DOS or **NETPRINT** from Netware) is resident on the system, **TwinCAD** will continue to prompt:

Queue it to the Printer Spooler? Y

asking whether to send the file automatically to the printer spooler. **TwinCAD** will report "--OK!" or "-- Not OK!" to reflect the status.

The output file extension is fixed to "**PRT**". The printer spooler will not erase it automatically after the printing. The operator must do the house-keeping job by himself.

## Limitations and Hints

The following describes some of the printing limitations you should know about.

### Maximum Printing Width and Printing Length

As the printer driver supports only integer arithmetic, **TwinCAD** must pass all the parameters in integer form. The values of the printing width and length are converted into unsigned integers with an integer unit representing a real unit of **1/1000**". Apparently, the maximum representable value is 65.535 inches. **TwinCAD** will not check if a setting value is larger than this maximum value, and the wrap-around may result when it happens.

Another limitation lies in the fact that the driver must convert the real length value into equivalent number of dot columns and rows, which is also a 16-bit unsigned integer. If the printing resolution is larger than 1000 *dpi*, say 1200 *dpi*, the maximum printing width and length will be further limited to the value  $65535/1200 = 54.6$  inches.

The printer driver of current release allocates 64KB of real mode memory for use as the graphic image band output buffer. Depending on the printing width required, this buffer will be sub-divided into several image bands of the required dot columns in length to output. However, the height of this band will depend on the type of the printer, and the printer driver requires that at least one such image band can be set up in the output buffer to meet the output requirement. This sets further limitation on the maximum printing width.

For example, a 48-pin nozzle ink-jet printer may require the graphic image to be sent in 48-dots in height. Suppose that the maximum effective pen width in use is 8 dots (which means 8 dot rows must be used for overlapping in image generation), the maximum width that can be printed with only one image band in the buffer is  $65536 \cdot 8 / 56 = 9362$  dots. If the printing resolution is in 360 *dpi*, then the maximum width will be  $9362 / 360 = 26$  inches.

Of course, the printer may impose further physical limitations on the maximum printing width and length.

## The Pen Width and Its Resolution

The maximum pen width that can be generated by the printer drivers of current release is 24 printing dots. Depending on the printing resolution, the maximum pen width and its resolution in real unit can be calculated. For example, if 300 dpi is used for the printing resolution, the maximum pen width will be 24/300 inch, which is equivalent to 2.03 mm. The pen width resolution starts from 0.085 mm (1 dot) and also steps in 0.085 mm. The driver will convert the pen width from the real unit to the nearest pen width in units of printing dot.

## Printing Speed Consideration

The printing time is determined by the graphic image generation and the image data I/O. The image generation requires the driver to process the graphic display list and generate the image to the output buffer. The image processing time depends on the length and complexity of the display list. The number of processing pass depends on the total amount of the generated graphic data to output, using the output buffer.

So, the printing speed will be dominated by the following factors:

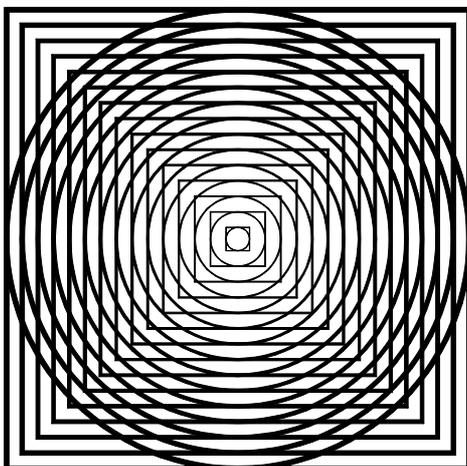
- **The printing resolution.** Higher resolution means more graphic data to generate, thus slower printing speed.
- **The printing area.** Larger printing area means more graphic data to generate, thus slower print speed.
- **The pen width in use.** The larger the number of lines to be printed with a wide pen (other than a single dot) and the wider the pen in width, the longer the image processing time.
- **The number of vectors in display list.** The more number of the display vectors in the list, the longer the image processing time.

## Procedure:

```
@CMD: PRPLOT (return)
What to plot -- Display, Extents, Limits or Window <D>: (return)
(Window pop up)
...
```

## Example:

See next page for an example of different pen widths in print-out.



# PTRIM

## Profile Trimming Command

---

### Purpose:

The **PTRIM** command is used to trim objects successively so that a profile can be quickly isolated and defined.

### Description:

The **PTRIM** command, used when a profile result is intended, lets you trim objects successively, so that a profile defined by these objects can be quickly generated. During the **PTRIM** operation, **TwinCAD** performs internally successive **FILLET**s with zero radius. Each time you pick up two objects, **TwinCAD** will trim them at their point of intersection which is determined on where the points are picked (see **FILLET** command). However, for successive profile trimming, the latter one picked (the picked point as well as the object itself) in the object selection pair will become the first one for the next selection pair, so all you need to do is to select the next object to trim.

To start **PTRIM** on a group of objects, you may pick on any one of its segments, followed by the next one, and continue on the operation successively until the whole profile is clearly trimmed. During the trim operation, if you pick up the last-picked and trimmed object again, nothing will happen except that the picked point of the object will be updated. This function is helpful since it is the position of the picked point that determines the result of trimming.

You may undo the last trimming by entering the sub-command option "**U**".

To exit the operation, reply to the prompt with a null return or type Ctrl/C.

### Special Notes

Since the **PTRIM** issues **FILLET 0** to fillet the successive objects, all rules imposed on **FILLET** command are also applicable to **PTRIM** command, except the following:

1. If the first two objects selected for the trimming are the opposite end segments of an open POLYLINE, the polyline will be trimmed directly and made closed.
2. In the midst of **PTRIM** operation, if the next selected object is a segment of the same polyline containing the last selected object, then the trimming will be performed if and only if both segments intersect with each other. And the newly selected object will become the last selected object.
3. If the two selected segments belong to different polylines, then they will be trimmed and the two polylines will be joined together to form one single polyline.
4. If a **PTRIM** operation makes a closed polyline, then the **PTRIM** command will exit automatically after the operation.
5. The **PTRIM** will not automatically convert single line or arc segments into POLYLINE, nor will it add them to an existing polyline.

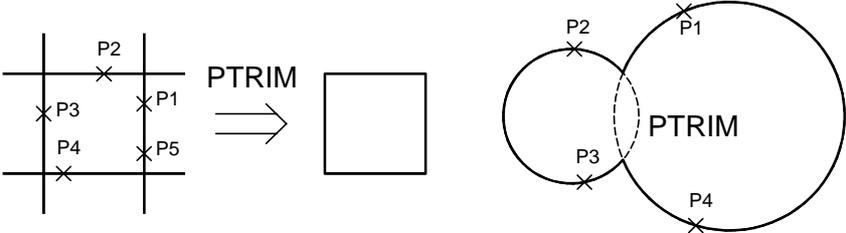
### Procedure:

To trim out a profile, type **PTRIM** to the command prompt, as the procedure below:

@CMD: **PTRIM** (*return*)

Select consecutive edge to trim: *(pick starting one)*  
Select consecutive edge to trim: *(pick next one)*  
...  
Select consecutive edge to trim: *(pick next one)*  
Select consecutive edge to trim: *(return)*

**Example:**



**'PUCS'**

## Projected-UCS-plane Axes Manipulation Command

---

**Purpose:**

The **PUCS** command is used to manipulate the Projected-UCS.

**Description:**

The **PUCS** command provides general manipulation of the Projected-UCS.

When you enter the **PUCS** command, **TwinCAD** will prompt

```
PUCS -- Save/Restore/ROtate/Axoplane/OFF/<base:(xxx,yyy)>:
```

where (xxx,yyy) is the current PUCS base point. In response, you may enter the following options:

- S**     **Save**, request to save the current Projected-UCS-plane axes setup with a name, so that it can be retrieved directly with the "R" option. You will be asked to supply the name.
- R**     **Restore**, request to restore the previously saved PUCS setup and use as the current Projected-UCS-plane axes setup. You will be asked to supply the name of the previously saved PUCS to restore from. If you type "?" to the prompt, **TwinCAD** will list out all the names of saved PUCS.
- RO**    **Rotate**, request to rotate current Projected UCS-plane axes setup around the virtual Z-axis by a specific angle. Note that the rotation is made by changing the direction of the current X axis and Y axis (as specified by the **SNAPANGLE** and **SNAPBETA**) by an angle measured in the projected plane. The specification for the projected Z-axis is not a concern. Note: When an Axonometric Projection plane has been set up, you may use **<Ctrl/I>** function to switch the projection plane to the desired one before doing the axis rotation. The **Rotate** option is especially useful in building skew PUCS-planes.
- A**     **Axoplane**, request to invoke the **AXOPLANE** command to prepare for an Axonometric Projection plane axes setup.
- <pnt>**   **Point designation**, set the new PUCS base coordinate.

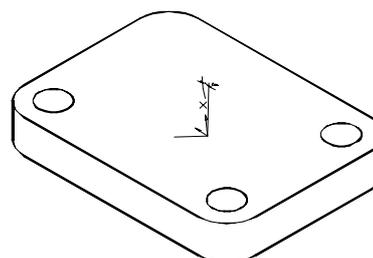
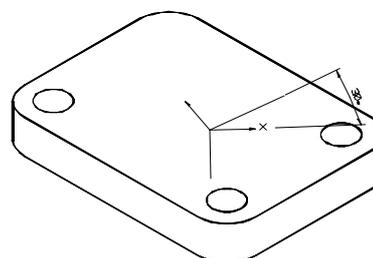
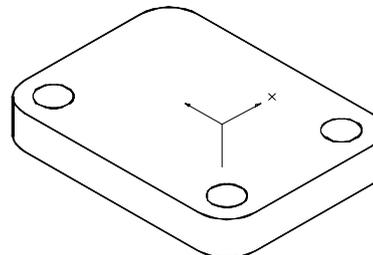
**Procedure:**

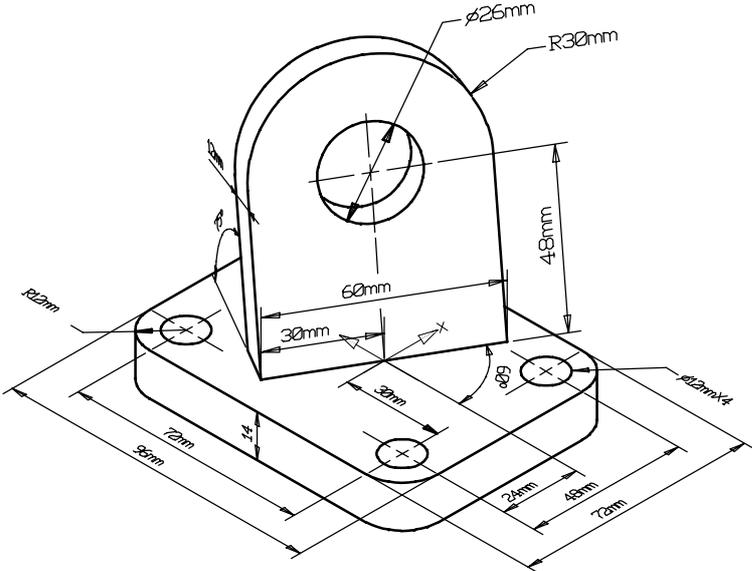
The following lists out an example procedure to set up two skew planes with respect to current plane setup.

```

...
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(43.301,45.)
>: cen of (pick ellipse)
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(43.301,57.)
>: 24,30
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)
>: RO
Angle value to rotate Projected UCS X-Y Axes: -30
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)
>: S
Save Current Projected UCS axes setup to: PLANE1
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)
>: RO
Angle value to rotate Projected UCS X-Y Axes: 15
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)
>: S
Save Current Projected UCS axes setup to: PLANE2
PUCS --
Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)
>:
...
PUCS -- Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)>: R
Load Current Projected UCS axes setup from (?): PLANE1
Override Current Base Point? <Y>
PUCS -- Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)>:
...
PUCS -- Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)>: R
Load Current Projected UCS axes setup from (?): ?
PLANE1      Axes=<114.90,106.10, 16.10>, Unit=<1.1971,0.7530,1.0000>
            Org: < 38.105, 84.0000>, Scale=1.22474487
PLANE2      Axes=<127.37, 33.26,222.06>, Unit=<1.1456,0.5045,1.1971>
            Org: < 38.105, 84.0000>, Scale=1.22474487
Load Current Projected UCS axes setup from (?): (space bar)
PUCS -- Save/Restore/ROtate/Axoplane/OFF/<base:(38.105,84.)>: ^C (type Ctrl/C)
...

```





# PURGE

## Removing All Unused Named Objects Command

---

### Purpose:

The **PURGE** command is used to remove all un-referenced named objects from the drawing.

### Description:

The **PURGE** command lets you do the house-keeping job by removing all the unused Blocks, Text styles, Layers and Regions. It also refine the drawing data structure by putting logically connected objects (like Block, Polyline, Region and so forth) together physically, thus making the disk access faster.

You may choose to purge all or some selected objects by entering the sub-command options:

- **B** -- Block
- **T** -- Tag Definition
- **LT** -- Linetype
- **LA** -- Layer
- **ST** -- Style
- **SY** -- Symbol
- **ALL** -- All of them (default)
- **Q** -- **Query mode**, request **TwinCAD** to prompt for each unreferenced Layer, Style, Block and Linetype for confirmation before purging it. Note that the anonymous block will not be queried even in the query mode. And if you reply to the query prompt with <Ctrl/C>, **TwinCAD** will skip the query mode for all the same kind of objects.

The purging operates with one option at a time. **TwinCAD** will not start purging until you reply to the prompt with a null return. **TwinCAD** continues to prompt you for other options. See Procedure for example.

You may purge the drawing at any time. The only drawback is that once a **PURGE** is done, you can not undo the process. The **UNDO** buffer will be cleared after a **PURGE** command is executed. **TwinCAD** will ask for confirmation before purging the drawing.

Note that **TwinCAD** refines the drawing database by saving the whole drawing to a temporary disk file, re-initializing **TwinCAD**, and restoring it back.

**Warning:** IF THERE IS ANY DISK I/O ERROR WHICH MAY HAPPEN, YOUR DRAWING WILL BE LOST.

### Procedure:

To remove unused linetypes and layers from the drawing, follow the example procedure below:

```
@CMD: PURGE (return)
Purge -- Block/Tag/LType/LAyer/STyle/SYmbol/ALL<ALL>: LT
Purge -- Block/Tag/LType/LAyer/STyle/SYmbol/ALL<LT>: LA
```

Purge -- Block/Tag/LType/LAyer/STyle/SYmbol/ALL<LA,LT>: *(return)*

Warning! The result of PURGE will not be recoverable.

The Undo buffer will be cleared. Proceed the operation ? <N>

**Example:**

# QBREAK

## Quickly Break Entity Command

---

### Purpose:

The **QBREAK** command is used to break a selected line or arc (or elliptic-arc) into two parts, or a circle (or ellipse) into a 360° arc.

### Description:

The **QBREAK** command lets you quickly break a selected line or arc (or elliptic-arc) into two parts, or a circle (or ellipse) into a 360° arc.

You will be asked to select the object to break. Once you have picked up the object, it will be separated into two parts at the picked position. If the selected object is a circle, it will be transformed into a 360° arc with the start point (and end point too) at the picked position. Likewise, an ellipse will be broken into a 360° elliptic-arc.

If the selected object is a segment of a POLYLINE, the object will also be broken into two parts, but **the joining relationship will be maintained**. The polyline will not be separated into two polylines, yet a new vertex is added at the point of break. This feature is useful in many applications (e.g., CAM applications).

Valid objects for this operation are LINES, ARCs, ELLIPSEs, CIRCLEs and POLYLINEs only.

You have to type <Ctrl/C> or reply to the prompt with a null return to exit the command.

### Special Note

The **QBREAK** command recognizes the picked point snapped by the **INT**ersection directive and breaks both entities at their point of intersection.

### Procedure:

Follow the procedure below to quickly break an object into two parts:

```
@CMD: QBREAK (return)
Select object to break: (pick one)
...
Select object to break: (return)
```

### Example:

# QCHANGE

## Quickly Change Entity Span Command

---

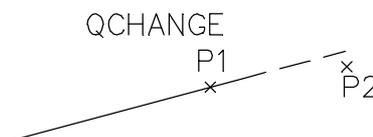
### Purpose:

The **QCHANGE** command is used to modify the span of a selected line or arc, or the radius of a circle.

### Description:

The **QCHANGE** command lets you quickly modify the length of a line or the span of an arc by extending or breaking it, or change the radius of a circle.

You will be asked to select the entity (line, arc or circle) first, and indicate the point of change to it. If the point of change is outside of the entity (in terms of the span of the entity), then the entity is extended to the change point. Otherwise, the entity will be trimmed at the change point and the portion containing the first selected point will be retained.



If the selected object is a Line or an Arc entity, you may use the object snap directive **PER** to snap at an existing object such that the Line/Arc will be trimmed or extended to a point on that object if applicable. You may also enter the length value of the entity directly (relative or absolute length, negative or positive value). This length adjustment will modify the nearest end point of the entity to the pick point.

If the selected object is a circle, then the radius of it will be changed such that the new circle will pass through the change point. Dragging of the circle is active when you are indicating the point of change. The desired radius value can also be entered directly from keyboard.

The **QCHANGE** command will continue itself until you reply to it with a null return or <Ctrl/C>. A sub-command option "Undo" is provided for you to undo the last change.

This command is helpful when you want to extend a line or an arc entity without specifying boundary entities.

### Additional Features

Additional features had been added to **QCHANGE** command after V2.0 to facilitate the editing operation, as described in the paragraphs below.

#### Change Text Contents

You may use **QCHANGE** command to edit the text content of a selected TEXT entity directly without changing its other properties. This will save you a lot of operation time compared with using **CHANGE** command.

You may also use **QCHANGE** command to edit the text content of a local tag attribute directly.

A text entry field will be opened at command area for you to edit the text content.

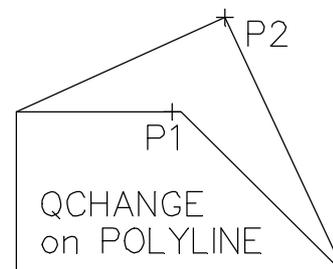
## Change Vertex Position of a Polyline

You may use **QCHANGE** command to move a selected vertex of a polyline to a new position by picking at its nearby segment. **TwinCAD** will prompt

Indicate point to change:

asking you to designate the new position of the vertex. The position change of a vertex will modify the geometry of its adjacent segments by the following rules:

- One end point of the segment is changed to the new location of the moved vertex.
- If the segment is an arc, the new arc will pass through the midpoint of the original arc segment.



A dragging function to show the change of its adjacent segments is provided.

## Change Leader Dimension

You may use **QCHANGE** command to change selected node positions of leader dimensions. Pick up a leader dimension by one of its node points, and designate a new position for it. The selected node point will be changed to the new designated position.

A dragging function to show the change of the leader line/spline is provided.



## Change Center Dimension

You may use **QCHANGE** command to change the size of a Center Dimension. You may change the size of the center mark or the size of the outreaching arms, depending on which part of it was picked on. If you pick on the arm, the change point will modify the length of the four arms. If you pick on the center mark, the change point will change the size of the mark.

## Change Linear Dimension

You may use **QCHANGE** command to modify an existing Linear Dimension by the following rules, depending on which part of the dimension is picked (nearest definition point to the picked point):

- If the text insertion point is the nearest one, the text insertion point will be changed to the new point.
- If either of the dimension base points is the nearest one, the dimension line will be extended or trimmed according to the change point, i.e., the dimension base point will be changed to the new point. However, this change will not affect the measurement direction.
- If either of the dimension extension points is the nearest one, it will be changed to the new point such that an oblique effect is produced. The text insertion point will also be updated to keep its original topological relationship with the dimension line. Note that if the dimension was created by HDIM or VDIM, it will be changed to ALDIM rotated with respect to either horizontal or vertical direction. Once a dimension is in Rotated state, the oblique effect will be done only to modify the dimension line.

## Change Angular Dimension

You may use **QCHANGE** command to modify an existing Angular Dimension by the following rules, depending on which part of the dimension is picked (nearest definition point to the picked point):

- If the text insertion point is the nearest one, the text insertion point will be changed to the new point.
- If either of the dimension extension points is the nearest one, both of them will be changed such that the radius of the dimension line is equal to the distance from the change point to the angle vertex. The text insertion point will also be updated to keep its distance to the dimension line unchanged.
- If the end point (undefined point) of either dimension extension line is the nearest one, it will be changed to the new point. This is one way to modify the length of the extension line once the dimension is created.

### Procedure:

Follow the procedure below to quickly modify an object's length:

@CMD: **QCHANGE** (*return*)

Select LINE/ARC to quickly break/extend: (*pick one*)

Indicate point to break/extend: (*point*)

Select LINE/ARC to quickly break/extend: (*pick one*)

...

### Example:

'QTEXT

## Command

Quick Text display control mode ON or OFF.

Quick Text display control mode ON or OFF. When Quick Text displays will be replaced by rectangle boxes display.

Shows the status of this display control. This is an

variable "**AUTOQTEXT**" to control the Quick Text command.

M

# QTRIM

## Quickly Trim Object Command

---

### Purpose:

The **QTRIM** command is used to trim or extend an object more quickly.

### Description:

The **QTRIM** command lets you quickly trim or extend objects without going through tedious operation as in the **TRIM** or **EXTEND** command. See the **TRIM** and **EXTEND** commands for the differences.

This command requires only two pick-ups on objects to complete one trimming/extending operation. The first one is for the boundary object, and the second one for the object to be trimmed or extended. If the two objects intersect with each other, the pick-up portion of the latter one will be retained and the rest of the object beyond the point of intersection will be trimmed. If they do not intersect, however, the object (the second pick-up) will be extended to the point of intersection.

Note that only Lines, Arcs, Circles, Ellipses and Polylines can serve as the boundary objects. And only Lines, Arcs, elliptic-arcs and open polylines can be trimmed or extended by this command. This includes 3D-lines.

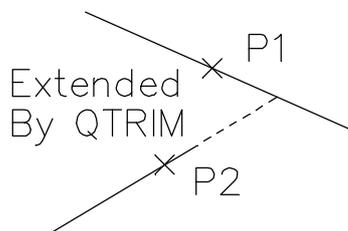
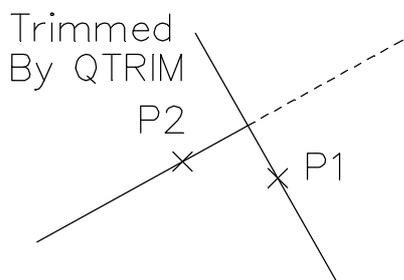
The **QTRIM** command will continue itself until you reply to it with a null return or Ctrl/C. You may also undo the last operation by entering the sub-command option "U".

### Procedure:

Follow the procedure below to quickly trim an object:

```
@CMD: QTRIM (return)
Select boundary edge: (pick one)
Select object section to extend/keep: (point)
...
```

### Example:



**QUIT**

## **Quit TwinCAD without Saving Work Drawing Command**

---

**Purpose:**

The **QUIT** command is used to leave **TwinCAD** without saving current work.

**Description:**

The **QUIT** command will leave **TwinCAD** without saving the current drawing to disk. You will be asked to confirm the request before the system exits, if the drawing has been changed since the last explicit saving or **NEW**/loading or **NEW**/creation.

**Windows Specific**

If no script is active when this command is issued, a Yes-No-Cancel message box will be pop up for the confirmation.

**Procedure:**

To quit current drawing and exit **TwinCAD**, type as below:

@CMD: **QUIT** (*return*)

Really want to discard all changes to drawing? <N>: **Y** (*return*)

**Example:**



# RDIM

## Radius Dimensioning Command

---

### Purpose:

The **RDIM** command is used to create a radius dimension.

### Description:

The **RDIM** command lets you dimension an arc or a circle by its radius. You pick up an arc or a circle to dimension, and **TwinCAD** generates the radius dimension for you. Dragging function is provided for you to decide where to place the dimension, and is started as soon as you pick up the arc or circle.

The dimension text is generated automatically in the drawing unit to reflect the true radius value of the object. The generation of this default dimension text is governed by the dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about dimensioning.

The dimension style is determined by the position of the dimension text during the interactive dragging. The dimension line will be inside of the circle on the radius, if the dimension text is dragged inside, and will be outside if the text is dragged outside. The change can be seen directly from the dragging operation.

During the dragging operation, you may enter a direct angle value in unit of degrees to specify the direction angle of the dimension line required for the dimensioning. The dimension text will then be placed at about the same distance from the current dragging position to the center of the dimensioning.

If the accuracy of the angle direction of the dimension line is of no concern, you may simply designate a point to settle down with the placement of the dimension. The dragged image serves as a preview of the dimensioning when you designate the point by current cursor position.

Before settling down with the placement of the dimension, you may modify the dimensioning by the following sub-command options:

- D**     **Default text**, specifying to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about it.
- C**     **Change text**, specifying to change the dimension text manually. **TwinCAD** will prompt:

Dimension Text :

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

- OD**    **Dimension-Line Option**, specifying to toggle the current effect of dimension line option. If this option is ON, the dimension line will be longer than the radius and passing through the center while the text is at the end of the dimension line without

arrow pointer. If the text is outside, the dimension line will be extended across the diameter inside the circle and the arrow pointer is generated at the end of the dimension line pointing outward to the circle (inside of the circle). Note that if the **OR** option is ON, the effect of **OD** will be modified when the dimension text is outside of the circle. See also **OR** option.

This option can be preset at bit 0 of the dimension variable **DIMROPT**.

**OT** **Text generation option**, specifying to toggle the current effect of text generation option. It specifies whether to align the dimension text with the dimension line or not. If this option is OFF, the dimension text generated will always be horizontal. When it is ON, the dimension text will be generated in the direction as the dimension line. This option can be preset at bit 1 of the dimension variable **DIMROPT**.

If the dimension text is placed outside of the circle and this option is OFF, an additional lead line will be added to lead the horizontal text to the dimension line. The length of this lead line is determined by the dimension variable **DIMDRLEAD**.

**OA** **Text alignment option**, specifying to toggle the current effect of text alignment option. It specifies whether to align the dimension text above the dimension line or not. If this option is OFF, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. If it is ON, the text will be placed above the dimension line (and the dimension line will be extended to cover the span of text). This option can be preset at the bit 2 of the dimension variable **DIMROPT**.

**OR** **Dimension line reverse option**, specifying to toggle the current effect of dimension line reverse option, which is effective only when the dimension text is dragged outside of the dimensioned circle. This option is to be used with the **OD** option, and will have the effect of reversing the dimension arrow pointer direction or extending the dimension line depending on the status of the **OD** option.

**Case 1: OR ON and OD ON.** The dimension line spans from the center to the perimeter of the circle, with the arrow pointer inside of the circle and pointing toward the dimension text.

**Case 2: OR OFF and OD ON.** The dimension line spans across a diameter of the circle, with the arrow pointer inside of the circle and pointing away from the dimension text.

**Case 3: OR ON and OD OFF.** The dimension line extends from the perimeter to the center of the circle, with the arrow pointer outside of the circle and pointing toward the center.

**Case 4: OR OFF and OD OFF.** Both the dimension line and the arrow pointer are outside of the circle.

This option can be preset at the bit 3 of the dimension variable **DIMROPT**.

### Auto-extension-arc Feature

When you are dimensioning an arc with the dimension generated inside of the arc, and the dimension line points to a place which is out of the arc span, then the auto-extension-arc feature will be activated and an extension arc will be generated for dimensioning purpose. In fact, this auto-extension-arc feature will always be activated whenever the dimension arrow pointer points outside of the dimensioned arc.

### Dimensioning on PUCS-Plane

The **RDIM** command allows you to dimension a projected circle (Ellipse) on the Axonometric PUCS-plane from the virtual space directly. The dimension rules are the same as those for **DDIM** command. See also **DDIM** command for more information.

## Length of Dimension Text Lead Line

When the dimension text is placed outside of the dimension circle, an additional horizontal line is added to lead to the dimension text. The length of this horizontal lead line is determined by the system variable **DIMDRLEAD**. If the setting value is positive and non-zero, it specifies the absolute length value of the lead line in the drawing unit. If the value is negative, it specifies the relative size in ratio to the current arrow size in use. If it is zero, which is default and compatible to the previous setting, it is equivalent to the setting of -1, which specifies to use the current arrow size as the default length.

Note that this variable affects only the user interface in the creation and modification of the dimension entity.

### Procedure:

Enter the **RDIM** command to **TwinCAD** command prompt:

@CMD: **RDIM** (return)

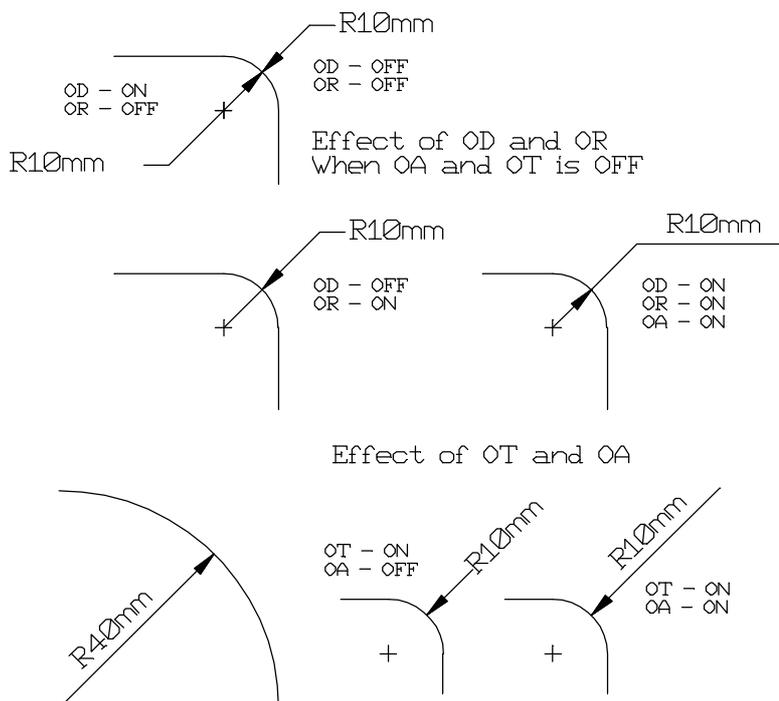
and **TwinCAD** will prompt in sequence as below:

Select arc/circle: (pick one)

Default-text/Change:<xxx>/OT(Text option)/OR/OD(Dimension-line option)/

OA(Adjust-option)/<Set dimension text position or direction:(point or value)

### Example:



**'RECORD**

## Set History Recording Command

---

**Purpose:**

The **RECORD** command is used to control the Operation History Recording function.

**Description:**

The **RECORD** command lets you turn the Operation History Recording function **ON** or **OFF**, save the current recording content to disk file, or clear all the history record stored in memory.

The following sub-command options are available:

- **ON** - Request to turn the recording function ON.
- **OFF** - Request to turn the recording function OFF.
- **C** - Clear, request to clear the recording memory.
- **S** - Save, request to save the recording buffer from memory to disk. A file window will pop up for you to specify the output record file. The file extension of this output record file will always be "**REC**".

You can type **<Ctrl/Z>** to review the operation history record, or type **<Ctrl/S>** to toggle the state of recording function ON/OFF.

The Operation History Recording may take a large amount of disk space if it is enabled after a long period of operation. It is recommended to clear it as it grows big. Turn it off when you do not need the information of the operation history, and turn it on when you want to list something.

**Procedure:**

To control the History Recording function, follow the procedure below:

@CMD: **RECORD** (*return*)

Set RECORDing ON/OFF/Clear/Save <ON>: (*enter options*)

**Example:**

**REDO**

## Undo the Last UNDO command

---

**Purpose:**

The **REDO** command is used to undo the Last **UNDO** command, i.e., to restore the drawing state before the last **UNDO** command.

**Description:**

The **REDO** command lets you recover the drawing state from the last **UNDO** command.

You may execute this command for an unlimited number of times as long as there is a preceding **UNDO** command to be recovered from and there is no other commands entered in between. For example, after you have issued the **UNDO** command twice, you may issue the **REDO** command twice as well to recover the drawing back. However, if you have issued, say, a **LINE** command after the two **UNDO** commands, you will not be able to recover from the previous two **UNDO** commands, and an error message will appear:

No UNDO informations left, can't REDO now!

This error message will also appear when there is nothing to **REDO**. Meanwhile, **TwinCAD** will automatically give you the message:

This is the last REDO.

when the last **UNDO** state is recovered.

See also **UNDO** command.

**Procedure:**

To recover the last **UNDO** state, type as below:

@CMD: **REDO** (*return*)

**Example:**

**'REDRAW**

## Redraw the Screen Command

---

**Purpose:**

The **REDRAW** command is used to refresh the drawing in the graphic screen.

**Description:**

The **REDRAW** command lets you refresh the drawing in the graphic screen.

You can also refresh the drawing in the graphic screen by typing **CTRL/N** to tell the **TCAM Graphic Runtime** to refresh the screen with the graphic display list.

**Procedure:**

To refresh the graphic screen, type as below:

@CMD: **REDRAW** *(return)*

**Example:**

**'REGEN**

## Regenerate the Drawing Command

---

**Purpose:**

The **REGEN** command is used to regenerate the drawing to the graphic screen.

**Description:**

The **REGEN** command lets you regenerate the drawing from the drawing database to the graphic screen (graphic display list). You can reach the same function by typing **CTRL/R**.

You may interrupt the drawing regeneration by pressing <Ctrl/C>.

**Procedure:**

To regenerate the drawing, type as below:

@CMD: **REGEN** (*return*)

**Example:**

# REGION

## Create A Region Entity Command

---

### Purpose:

The **REGION** command is used to define a region entity.

### Description:

The **REGION** command lets you define a region entity. A region entity is a named entity, like a Block entity, and is referenced by name. A region is a closed area bounded by circles, closed polylines or other region entities. Therefore, a region entity is defined by a group of circles, closed polylines or other region entities. You may define a region entity in the same way as you define a Block.

Like an ordinary entity, a region entity can be duplicated, erased, rotated, scaled and moved. Moreover, a region entity can assume an imposed state, such as solid color fill (**FILL**), pattern hatch (**RHATCH**), .. etc. The imposed state of a region may generate special effect for the region, but will never take additional space of the entity until it is exploded.

The properties and operations about a region will be further explored in the future release of **TwinCAD** for applications in industries. Currently, it is mainly used to define an area of solid color fill.

When you issue the **REGION** command, **TwinCAD** will pop up a slide window with region name selection for you to enter the name of the region. If you select an already defined region name, **TwinCAD** will quit the command. To define a new region, you have to pick up the **<NewRegion>** item from the window, and enter the name for it.

The next step for you to do is to select the closed objects which define areas. Once the selections are done, a region is defined. Note that you can not directly access the entities which make up the region, just like you can not access the one in a block instance created by an **INSERT** command. This will be changed in future release.

### Procedure:

To define a region, follow the procedure

@CMD: **REGION** *(return)*

Enter region name to define: *(Operate on pop-up window)*

**TwinCAD** pops up a slide window, and you must select the **<NewRegion>** item and enter the name for the new region.

Select Objects (+): *(Do so, after object selection, region is defined)*

### Example:

See example drawing of regions given in **FILL** command.

**REMOVE**

## Remove Specific-Named Objects Command

---

**Purpose:**

The **REMOVE** command is used to remove selected objects from the drawing.

**Description:**

The **REMOVE** command lets you remove selected objects from the drawing, including Layer, Linetype, Block, Region, Attribute Tag and grouped Tags, based on the sub-command options further specified.

When you enter the REMOVE command, **TwinCAD** prompts:

LAyer/LType/Block/Region/Tag/Grouped-tags:

The sub-command options provided are as below:

- LA**     **Layers**, request to remove specific Layers.
- LT**     **Linetype**, request to remove specific Linetypes.
- B**       **Blocks**, request to remove specific Block definitions.
- R**       **Regions**, request to remove specific Region definitions.
- T**       **Tags**, request to remove specific Tag variables.
- G**       **Grouped Tags**, request to remove specific grouped Tag variables.

Once a sub-command option is specified, **TwinCAD** will pop up a dialogue window containing a selection slide sub-window with the list of the names. You may selectively mark or unmark a name by picking at it. If a name is marked, a '#' sign will be prefixed to the name. After the selection, you may pick up **[OK]** button to confirm the operation and exit the command. To quit the operation, you may simply pick up **[-]** or **[Cancel]** button.

An additional button labeled as **[All-Item]** also appear in the dialogue window. You may pick up this button to toggle the selection state for all the items in the slide sub-window.

The following paragraphs describe more details about the removal of each different type of the named object.

### Removal of Layers

You can remove all the layers except for Layer "0". If the current layer is removed, Layer "0" will become the current layer. All entities on the removed layers will be moved to layer "0" and may be selected directly from the Previous Selection set immediately after the **REMOVE** command.

### Removal of Linetypes

You can remove all the linetypes except for the "CONTINUOUS" linetype. If the current linetype is removed, it will be changed to BYLAYER. If the linetypes referenced by a layer are removed, then the linetype control for the layer will be changed to "CONTINUOUS". Likewise, if a linetype referenced by an entity is removed, then the linetype control for the entity will be changed to BYLAYER. All modified entities may be selected directly from the Previous Selection set immediately after the **REMOVE** command.

## Removal of Blocks

You may remove all the block definitions. Please note that when a block definition is removed, all the block instances (INSERT) of that block will also be erased. A special mark '!' will appear before the name of a block in reference, which reminds you that removing the block will also erase something from the drawing.

All unreferenced anonymous blocks will be marked and removed automatically when the dialogue window is entered.

## Removal of Regions

The rules for removing regions are the same as those for the Blocks. See the previous paragraph.

## Removal of Tag Variables and Grouped Tag Variables

You may remove all the tag variables from the drawing. However, the removal of a tag variable will not affect the tag attribute entities. If a tag variable is removed, it will also be removed from the list in a Grouped Tag variable.

Note: Applying the **REMOVE** command is the only way to pull out unwanted TAG variables or Grouped TAG variables from a drawing.

## Special Note

The **REMOVE** command does not support the removal of a text style in current release. However, you may use **PURGE** command to remove unused text style.

## Procedure:

```
@CMD: REMOVE (return)  
LAYER/LType/Block/Region/Tag/Grouped-tags: (enter option)  
...
```

# RENAME

## Change the Name of Named Object Command

---

### Purpose:

The **RENAME** command is used to change the name of named objects in the drawing.

### Description:

The **RENAME** command lets you change the name of named objects in the drawing, including Layer, Linetype, Block, Region, Attribute Tag and Grouped Tags, based on the sub-command options further specified. The sub-command options provided are as below:

- **LA** -- Layers, request to change the name of layers. The layer control window will be popped up and allow you to directly change the name of layers. Note that the name of layer "**0**" will never be changed.
- **LT** -- Linetypes, request to change the name of linetypes. The linetype control window will be popped up and allow you to directly change the name of linetypes. Note that the name of linetype "**CONTINUOUS**" will never be changed.
- **B** -- Blocks, request to change the name of blocks. The block name selection window will be popped up and allow you to directly change the name of blocks.
- **R** -- Regions, request to change the name of regions. The region name selection window will be popped up and allow you to directly change the name of regions.
- **T** -- Tags, request to change the name of attribute tags. The attribute tag control window will be popped up and allow you to directly change the name of attribute tags.
- **G** -- Grouped Tags, request to change the name of grouped tags. The grouped tags control window will be popped up and allow you to directly change the name of grouped tags.
- **S** -- Style, request to change the name of text styles. The text style control window will be popped up and allow you to directly change the style name.

The operation of the name change may be canceled if you pick up the screen button [**CANCEL**] or the [-]. To confirm the change, you have to pick up the screen button [**OK**].

### Procedure:

@CMD: **RENAME** (*return*)

LAYER/LType/Block/Region/Tag/Grouped-tags/Style: (*enter option*)

...

**RESUME****Resume an Interrupted Script File Command**

---

**Purpose:**

The **RESUME** command is used to resume an interrupted command script execution.

**Description:**

The **RESUME** command is used to resume the execution of a command script that has been interrupted due to errors or keyboard input. The execution of a command script is invoked by the **SCRIPT** command. See **SCRIPT** command for details.

This is an ACAD-compatible command.

**Procedure:**

To resume an interrupted command script, type as below:

@CMD: **RESUME** *(return)*

**Example:**

# RHATCH

## Impose a Hatch State on a Region Command

---

### Purpose:

The **RHATCH** command is used to impose a hatch state on a region, such that a crosshatch pattern is generated over the region.

### Description:

The **RHATCH** command lets you crosshatch over a region. Unlike the **HATCH** command, the **RHATCH** does not generate real entities of hatch lines that take up the space of the drawing database, but imposes a hatch state on the region (specifying that the region has a crosshatching requirement.)

With this hatch state, the hatch lines over a region are generated on a temporary basis. These hatch lines are generated when they are defined by the **RHATCH** command over a region or when a region with a hatch state is loaded from the drawing. This saves the entity space and thus the storage of a drawing file, at the expense of taking more time to regenerate the hatch lines on loading the drawing file.

When you issue the **RHATCH** command, you will be asked to select a region to crosshatch. After the region is selected, the rest of procedures are the same as those for **HATCH** command after the object selection. The only difference is that the **RHATCH** command accepts at most 3 pass of hatch line specifications, while there is no such limit to the **HATCH** command.

You may use **EXPLODE** command to explode an **RHATCH** hatch lines to achieve the same result as it would be by issuing a **HATCH** command. See **HATCH** command for more details.

Another advantage of using **RHATCH** is the possibility of changing the hatch pattern with one single command or to produce the hatch pattern in terms of the output device. This will be implemented in the future release of **TwinCAD**.

### Procedure:

To impose a hatch state on a region, follow the procedure below:

```
@CMD: RHATCH (return)
Select Region: (pick one)
<Base point of hatch line>: (point or space bar)
<Hatch direction/Angle>: (value)
Hatch distance: (value)
(TwinCAD generates hatch lines)
<Base point of hatch line>: (^C)
```

### Example:

# RINSERT

## Create Polar Arrays of Block Instances Command

---

### Purpose:

The **RINSERT** command is used to create a polar array of multiple block instances.

### Description:

The **RINSERT** command lets you create a polar array of block instances. It provides a very similar function to the **INSERT** command in creating block instances. However, instead of inserting only a single block instance, **RINSERT** will insert multiple instances of the block in a polar array pattern.

You can take this command as a combination of **INSERT** and **ARRAY/Polar** commands. The only difference is that the block instances thus created are grouped as one single composite entity. Informations related to the polar array patterns are part of this entity, and are not a temporary specification of the multiple copies.

When you enter **RINSERT** command, **TwinCAD** will prompt you to specify the name of the Block to insert, the base point, the scale factor and the rotation angle of the insertion. The procedure is quite the same as that for the **INSERT** command. See **INSERT** command for details.

After you have completed specifying a single block instance, **TwinCAD** will start asking you about the geometry of the polar array pattern as it does in the **ARRAY** command under the polar array pattern option. See **ARRAY** command for more details.

The prompting sequence for the array information will be briefly described below:

Center point of array:  
Number of items: (*space bar for zero default*)  
Angle to fill (+=CCW, -=CW) <360°>:

If the number of item given is less than zero or equal to 1, the command quits. Note that the item number can be zero, as you press the space bar, to indicate the need for angle increment between elements. However, if both the item number and the angle to fill are zero, the command quits.

If both the item number and angle to fill are not zero, the angle increment between elements will be calculated by the rules stated below:

If the angle to fill is  $\pm 360^\circ$ , the specified number of items will be evenly inserted around the circle; otherwise, the angle increment will be calculated as  $(angle\_to\_fill/(item\_number-1))$ .

If the item number is zero while the angle to fill is not zero, the system will prompt

Angle between items:

asking you to input the angle increment directly. The angle increment must not be zero, otherwise the command will quit. The sign of this angle increment is irrelevant, since it is the angle to fill that determines the direction. The item number will be calculated as integer part of  $(angle\_to\_fill/angle\_increment)$  plus one.

If the angle to fill is zero while the item number is not zero, then **TwinCAD** will prompt

Angle between items (+=CCW, -=CW):

asking you to input the angle increment directly. The sign of this increment value will determine the direction of the insert.

Finally, **TwinCAD** prompts

Rotate objects as they are copied? Y

and continues the rest of operations. Note that **TwinCAD** will not ask for the common reference point if the objects need not be rotated, since the block insertion point will be used for that purpose.

### Procedure:

To make a multiple insert of TESTBLOCK, an example procedure is given below: (See INSERT command section for more example procedures)

@CMD: **RINSERT** (*return*)

Block name or (?) <...>: **TESTBLOCK**(*return*)

Insert base point: (*point*)

X scale factor <1>: (*value*)

Y scale factor <default=X>: (*value*)

Rotation angle <0>: (*value*)

Center point of array: (*Point*)

Number of items: (*Value or Space bar*)

Angle to fill (+=CCW, -=CW) <360°>: (*Value or Space bar*)

If both the item number and the angle to fill is zero, the command quits. If only the item number is zero (by space bar), then **TwinCAD** asks

Angle between items: (*Value*)

Or, if the angle to fill is zero, **TwinCAD** asks

Angle between items (+=CCW, -=CW): (*Value*)

After these prompts, **TwinCAD** continues to ask

Rotate objects as they are copied? <Y>: (*Y or N*)

### Example:

# ROTATE

## Rotate Objects Command

---

### Purpose:

The **ROTATE** command is used to rotate selected objects.

### Description:

The **ROTATE** command lets you rotate selected objects about a designated point (actually about a line passing through the rotation base point and parallel to the Z-axis) by an angle.

When you issue this command, you will be asked to select the objects to rotate first. See **SELECT** command for object selection operation. After you finish the object selection, you will be asked to specify the base point for rotation.

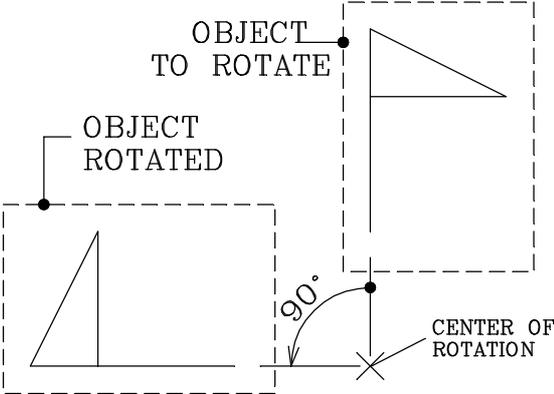
Once a base point is specified, you can drag the objects around the point while **TwinCAD** asks you to specify the angle of rotation. You may specify the angle of rotation by

- Designating a point. The angle rotated counter-clockwise about the base point from the positive axis to the designated point is taken as the rotation angle.
- Entering a value. The value is taken as the angle of rotation in units of degree.
- Entering option "**R**". This sub-command option will let you establish a reference angle, and then specify a new angle value relative to the reference angle to which the selected objects will be rotated. You may utilize the snap directive **DIR** to establish the reference angle.

### Procedure:

- To rotate objects by 45°, follow the procedure below:  
@CMD: **ROTATE** (*return*)  
Select Objects (+): (*do so*)  
Base point: (*point*)  
<Rotation angle>/Reference: **45**(*return*)
- To rotate objects by reference angle, follow the procedure below:  
@CMD: **ROTATE** (*return*)  
Select Objects (+): (*do so*)  
Base point: (*point*)  
<Rotation angle>/Reference: **R**(*return*)  
Reference angle <0°>: (*value*)  
New angle (°): (*value*)

**Example:**



**RSCRIPT****Restart a Script File Command**

---

**Purpose:**

The **RSCRIPT** command is used to restart the execution of a script file.

**Description:**

The **RSCRIPT** command lets you restart the execution of a script file from its beginning, provided it has been invoked and loaded just now. Usually, this command is placed at the end of a command script file for continuous demonstration purpose.

This is an ACAD-compatible command.

**Procedure:**

To restart a command script, type as below:

@CMD: **RSCRIPT** (*return*)

**Example:**

## Load and Execute a TCL Program

---

### Purpose:

The **RUN** command is used to load and execute a **TCL** program file.

### Description:

The **RUN** command lets you load and execute a **TCL** program file from the external disk. After the program exits, it will be unloaded automatically.



**SARC**

## Draw Spline Arcs Command

---

**Purpose:**

The **SARC** command is used to draw spline arcs.

**Description:**

The **SARC** command lets you create a spline segment approximated by two arcs. The spline segment is defined by a curve passing through two given points with given slopes at both ends.

When you issue **SARC** command to define a spline segment, you will be asked to designate the two end points (the start point and the end point) of the spline first. You can designate the two end points with the following point specifications:

- *An absolute point position*, such as the **END**point of an arc segment, **MID**dle point of a line segment, **CEN**ter point of a circle, or direct coordinates input, and so on.
- A **TAN**gent point to a circle/arc.
- A **PER**pendicular requirement (normal) to a specific object (line, arc and circle).

After the two end points have been specified, you will be asked to designate the third point, serving as a control point to the spline, to determine the slopes at the two end points of the spline. Once it is determined, the spline segment is drawn.

In **Standard Mode**, the slope from the start point to the control point defines the slope of the spline passing through the start point, and the slope from the control point to the end point defines the one passing through the end point. If in **Complement Mode**, the slopes will be taken in reverse direction. You may choose the standard mode or the complement mode by selecting the "**S**" or "**C**" sub-command options, respectively. The default is standard mode.

The starting slope may be determined by a directly given angle or by the direction continued from the last point of entity creation. You may specify the starting slope after the start point specification by entering the sub-command options "**D**" and "**DI**". The control point will define only the ending slope at its end point, if the spline arc has been constrained with an absolute starting slope at its start point.

The sub-command option "**D**" is used to specify the starting slope at an angle relative to a reference angle. If there is no object bound with the start point specification, this reference angle will always be zero. On the other hand, if the start point is determined on a given object (by **MID** of, **END** of, **QUA** of, **NEA**r to, etc.), the direction angle of the object at the point will be taken as the reference angle. This is also applied to the default start point specification, which is continued from the last line/arc segment created.

**TwinCAD** provides fast dragging capability in drawing a spline arc segment. When you are designating the second point, the system will let you drag a line from the start point to the cursor position if the starting slope is in free state. If the starting slope is already in constraint, you will be dragging an arc segment passing start point at the given angle.

The dragging line changes to spline arc segments controlled by the cursor movement, as soon as you finish the end point specification. So you can clearly see what the arc segments will actually result in advance.

Note that when the starting slope is constrained, the actual control point will be defined by the intersection of the two lines passing through the two end points with respective slopes. If there is no intersection, the resulting arc segments may be erroneous.

The following explains the possible sub-command options used in the operation:

- **DI** - Direction, request to specify the starting slope with an absolute angle of direction, appeared only after the start point specification.
- **D** - Deflection, request to specify the deflection angle of the starting slope relative to the reference angle, appeared only after the start point specification.
- **F** - Free, request to remove the constraint over the starting slope.
- **S** - Standard, request to generate the spline in Standard mode (Default).
- **C** - Complement, request to generate the spline in Complement mode.
- **L** - Line mode, request to switch to Line mode (**PLINE** command only)
- **A** - Arc mode, request to switch to Arc mode (**PLINE** command only)
- **CL** - Close, request to close the polyline under Sarc mode (**PLINE** command only)

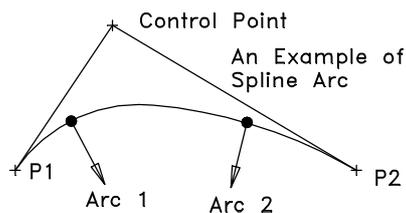
See also **PLINE** command for related informations.

Note: The actual arcs resulted from an **SARC** specification will depend on the setting of the system variable **SARCTYPE**. See also **SYSVAR** command.

### Procedure:

- **To draw a spline arc segment without starting slope constraint:**

```
@CMD: SARC (return)
@CMD: SARC <Start point of Sarc>: (point)
Direction/Deflect/Free/<End point of Sarc>: (point)
Free/Complement/Standard/<Control point>: (point)
```



- **To draw a spline arc segment with given starting slope:**

```
@CMD: SARC (return)
@CMD: SARC <Start point of Sarc>: (point)
Direction/Deflect/Free/<End point of Sarc>: DI (return)
Direction angle from start point: (value)
Direction/Deflect/Free/<End point of Sarc>: (point)
Free/Complement/Standard/<Control point>: (point)
```

- **To draw a spline arc segment in Complement mode:**

```
@CMD: SARC (return)
@CMD: SARC <Start point of Sarc>: (point)
Direction/Deflect/Free/<End point of Sarc>: (point)
```

Free/Complement/Standard/<Control point>: **C** (*return*)

Free/Complement/Standard/<Control point>: (*point*)

**Example:**

# SAVE

## Save Drawing to Disk Command

---

### Purpose:

The **SAVE** command is used to save current drawing to disk.

### Description:

The **SAVE** command lets you update the drawing on disk without exiting **TwinCAD**. It will pop up a file window for you to enter the output file. The current drawing file name will be the default choice. The file window may ask for confirmation again if you are to overwrite an existing file.

Note that if the current drawing is a DWG file, the default extension will be DWG, and if the current drawing is a WRK file, the default extension will be WRK. However, you may specify a file extension different from the default. Note that only when the file extension is in DWG will the drawing be saved to DWG file format.

After the saving, the name of the newly saved file will become the current drawing filename.

### Saving File with Encryption Password

If the system variable **ENCRYPT** is set to ON state, before saving the drawing, **TwinCAD** will ask you to enter an encryption string by the following prompt:

\*\* Caution!! Make sure you remember your encryption password!

Enter encryption string (password):

You may enter any character as the encryption string, as long as you can remember exactly what it is, including the upper/lower cases. The encryption string will be used to encrypt the output data string generated during the saving of the drawing to the disk.

If you reply to it with a null return, then the message

No encryption string, encryption disabled.

will appear and the drawing will be saved in the ordinary way.

Note that it is not possible to load an encrypted drawing file into **TwinCAD** without supplying the correct (original) encryption string. **ONCE YOU HAVE LOST THE MEMORY OF THE ENCRYPTION STRING, YOU HAVE LOST THE DRAWING FOREVER!**

### The Drawing Auto-save Feature

If you want to save the current editing drawing to the disk at each period of 'time', you may take advantage of the system variables "**AUTOSAVE**" and "**SAVETIME**", which are integer variables used to control the auto-save feature. When either of them is not zero, the auto-save feature is enabled and **TwinCAD** will save the current drawing to the disk at a specific interval automatically. When both of them are zero, the auto-save feature will be turned off.

This interval value mentioned above can be the number of effective commands that have been executed after the last disk saving, as being specified by the **AUTOSAVE** variable, or the time interval in minutes since the first effective command issued after the last saving operation, as being specified by the **SAVETIME** variable.

Note that the saving operation occurs only in the top level command processing when a command is finished and a new command is about to start. And, an effective command is a command that affects the drawing database and can be undone by **UNDO** command.

The output file saved by auto-save feature is specified by the system variable **SAVEFILE**. If it is a null string, the current drawing file will be used. If it is not a null string, it will be used as the name of the file to save to.

### The Auto-Backup Feature

If the system variable "SAVEBACKUP" is turned ON, **TwinCAD** will rename the existing WRK file to a backup file (BRK) before overwriting it with the newly saved version. However, for the same WRK file, the system will make backup only once at its very first saving operation. All subsequent savings to the same file either by **SAVE** command or by Auto-save feature will not create the backup file again.

So, you may issue the **SAVE** command as many times as you like to save the current editing drawing to the disk, with the very first old version of the drawing saved as the backup file.

#### Procedure:

To save current drawing to disk, type as below:

@CMD: **SAVE** *(return)*

File name <default>: *(Operate on file window)*

#### Example:

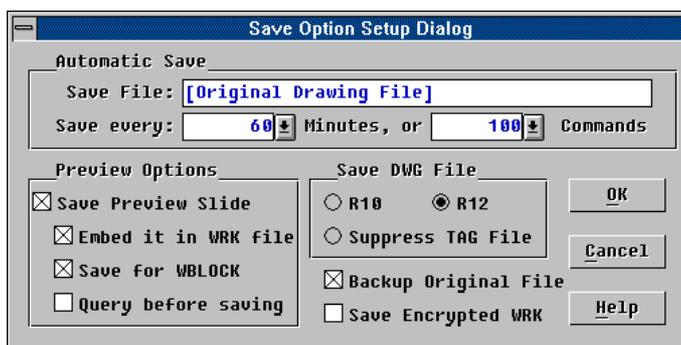
**'SAVEOPT****Drawing Saving Option Setup Command (TCL)****Purpose:**

The **SAVEOPT** command is used to setup drawing saving related options via a GUI dialogue operation.

**Description:**

The **SAVEOPT** command lets you access the drawing saving related options via a GUI dialogue window operation. These options, such as the AutoSave feature, saving backup file, creating preview image and DWG file version selection, are controlled by specific system variables. With **SAVEOPT**, you can easily access these options without knowing how to setup these related system variables.

**SAVEOPT** will pop up the dialogue window as shown below:



The following describe these screen items in details.

**Automatic Save**

The AutoSave feature will automatically backup the current drawing in editing session to the disk storage periodically, to prevent the possible loss of the work due to unexpected events. You may enable such feature from this group, which contains the following screen items:

**Save File** Filename entry, specifying the path name used for the drawing to be saved in the disk automatically. If a path name is explicitly given, the AutoSave feature will always backup the current drawing to the fixed file. If it is not given, AutoSave will save to its original file. You pick at this entry, and a file window will pop up for you to specify the path name. If you cancel the file window, then it means no explicit file is given.

**Save by time elapse** A combo-box of value entry and selection, specifying the time interval in minutes since the first undo-able command operation issued after the last drawing file saving operation that **TwinCAD** must save the drawing automatically.

If you pick at the value field, you will be asked to enter the value directly. Zero value means to disable the AutoSave feature based on this time elapse, and the value field will display "N/A". You may select a commonly used value from a drop-down list by picking at its drop-down button.

**Save by Command Count** A combo-box of value entry and selection, specifying the effective command counts, after which **TwinCAD** must save the drawing automatically.

If you pick at the value field, you will be asked to enter the value directly. Zero value means to disable the AutoSave feature based on the effective command counts, and the field will display "N/A". You may select a commonly used value from a drop-down list by picking it by the drop-down button.

An effective command is a command that affects the drawing database and that can be undone. Note that this will exclude the **UNDO/REDO** commands.

Note that the automatic saving operation occurs only in the top level command processing when a command is just finished or a new command is about to start. **TwinCAD** can not save the drawing if it is in waiting for the user input, even when the elapse time has passed the limit.

## Preview Options

When you are selecting files from the file window for a drawing loading operation, you may press the additional [Preview] button to view the drawing before loading it. The preview image of the drawing was actually created when the drawing was saved, not when it is put to view. If the drawing file does not contain any preview image, nor can an associated preview image file be found from the same path, you won't be able to see the preview image of it from the file window operation.

However, saving an additional preview image with the drawing requires additional overhead in the disk storage. So, **TwinCAD** does not compulsively save the preview image with the drawing. In fact, you may have the following options:

- **Save Preview Slide** -- You may choose to save a preview image in slide file format or not. In current release, **TwinCAD** create preview images only in slide file format, although it may preview the bitmap images as well. Check on this checkbox to enable **TwinCAD** to create a slide on the current drawing view when the drawing is saved.

Once you have enabled **TwinCAD** to create a preview image, you may have the following options:

- **Embed it in WRK file** -- If you are saving a WRK file, the preview image can be embed into the WRK file. If you choose not, or you are not saving WRK files (e.g. DWG file), the preview image will be saved as a separated image file of the same pathname in the disk, when the drawing file is saved.
- **Save for WBLOCK** -- You may choose to support the creation of a preview image for drawing saved by **WBLOCK** command. If this option is not enabled, **TwinCAD** will not create preview image for drawing saved by **WBLOCK** command.
- **Query before saving** -- You may choose to create the preview image based on individual query. If you check on this option, **TwinCAD** will pop up a query window for you to decide whether to create the preview image or not, whenever the drawing is saved.

## Save DWG File

**TwinCAD** can save DWG file in either R10 or R12 format. You may choose from this group to select the desired format. Also, **TwinCAD** will create a TAG file to save the tag definitions in the current drawing when the drawing is saved in DWG file format. If this is not desirable, you may suppress it by checking on the item "Suppress TAG File".

## Backup Original File

You may check on the item "Backup Original File" to enable **TwinCAD** to backup the original drawing file when the current drawing file is saved to overwrite it. If this option is enabled, the original drawing file will be renamed with BRK (for WRK file) or BAK (for DWG file) file extension.

## Save Encrypted WRK File

You may check on the item "Save Encrypted WRK" to enable **TwinCAD** to encrypt the drawing file when it is saved in WRK file. If this option is enabled, you will be asked to define a password string to encrypt the drawing. See also **SAVE** command for more details.

## Other Buttons

After completing your option setup, you have to press the **[OK]** button to confirm the change and terminate **SAVEOPT**. Or, you may press the **[Cancel]** to quit **SAVEOPT** operation. You may also press <ESC> key or the right mouse button to quit the operation.

## Special Notes:

The **SAVEOPT** command is an external command provided by the TCL program file "SAVEOPT.TCL" or "SAVEOPT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **SAVEOPT** command, you may solve the problem by copying the "SAVEOPT.TCL" to the COMMANDS sub-directory.

## Procedure:

Enter the **SAVEOPT** command to the system command prompt:

```
@CMD: SAVEOPT  
...[Dialogue Operation]...
```

# SCALE

## Scale Objects Command

---

### Purpose:

The **SCALE** command is used to change the size of selected objects by a scale factor.

### Description:

The **SCALE** command lets you change the size of selected objects by a scale factor. You will be asked to select the objects to scale via the object selection operation. See **SELECT** command for details.

After the object selection, you will be asked to designate the base point for the scaling and input the scale factor, as **TwinCAD** prompts in sequence:

Base point:

Reference/<Scale factor>:

You may input the scale factor by a negative value, but never a zero. A negative value will scale the objects with a mirror effect with respect to the base point. Note that **TwinCAD** will not drag the selected objects with a scaling effect, since the determination of the possible scaling factor upon the position of the cursor can not be practically devised at this point.

### Scaling by Alignment Point

In fact, if you designate a point to the prompt, **TwinCAD** will take it as a second reference point to align the objects after scaling. **TwinCAD** will then prompt:

To Point:

asking for the destination point where the given data point should be aligned to. Dragging of selected objects with scaling and rotation effect is provided, so that you may understand the purpose of the operation. The scaling with such point alignment may have a rotation effect other than the pure scaling over the selected objects.

Note that if you enter a value to this prompt, it will be taken solely as the scale factor.

### Scaling by Reference

The sub-command option "**R**" is provided for you to specify the scale factor by entering a reference length and a new length value to which the reference length will be changed. The scale factor is determined by the ratio of the two values. You may take advantage of the snap directives such as **LEN** and **RAD** to obtain the size of the object as a reference value, and then specify the actual size you need to scale the object, without calculating the actual scale factor required.

### Special Note

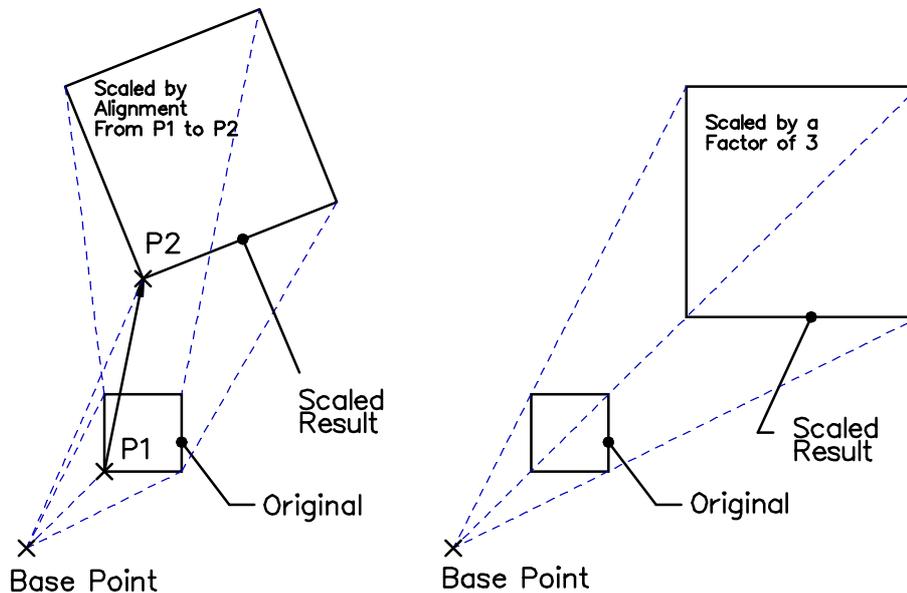
As the **SCALE** command modifies the geometry data of the selected entities directly, it is not possible for it to apply different scale factors (X-scale and Y-scale factor). However, if you need to scale an object by different scale factors in the X-axis and Y-axis direction, you may follow the following procedure to work out the job:

1. Use **WBLOCK** command to write out the selected objects to a temporary WRK file, specifying the scaling base point as the insertion base point.
2. Erase the selected objects by **ERASE/Previous** command.
3. Insert the written out WRK file into the drawing at the base point by using the **INSERT** command. Specify the name of the insertion as "\*=wrk-name", which means to merge it in without building the block. You may specify the different X and Y scale factors.
4. Use **DOS** command to erase the temporary WRK file.

### Procedure:

- **To scale objects such that the length of a line changes to 10:**  
 @CMD: **SCALE** (return)  
 Select Objects (+): (do so)  
 Base point: (point)  
 Reference/<Scale factor>: **R** (return)  
 Reference length <1.>: **LEN** of (pick the line)  
 New length: **10** (return)
- **To scale objects with a given scale factor:**  
 @CMD: **SCALE** (return)  
 Select Objects (+): (do so)  
 Base point: (point)  
 Reference/<Scale factor>: (value)

### Example:



**'SCOLOR**

## Set Symbol Local Colors Command

---

**Purpose:**

The **SCOLOR** command is used to set the local colors for use in symbols.

**Description:**

The **SCOLOR** command lets you assign the color numbers to each local symbol color. The local symbol colors define the colors assumed by the symbol elements (parts of a symbol) that have specification of local colors when they were built.

A symbol in **TwinCAD** is not monochromatic. It can possess several colors in its parts, besides the default one assigned to its insertion. These colors are local colors for the symbol. Each of them can be assigned with the real color number in the drawing colors. The initial default color assignment of these local colors are [**By Layer**].

When you issue the **SCOLOR** command, a symbol local color control window will be popped up. To assign new color number to a specific local symbol color number, pick up the corresponding item from the window. **TwinCAD** will open the color selection window for you to select the desired one. To exit the window, type <ESC> or press the rightmost button of the mouse or pick up the [-] screen button.

See also **SYMBOL** command for related information.

**Procedure:**

@CMD: **SCOLOR** (*return*)  
(*Operation on pop-up window*)

**Example:**

**SCRIPT**

## Execute a Command Script File

---

**Purpose:**

The **SCRIPT** command is used to execute the **TwinCAD** command from an ASCII text file.

**Description:**

The **SCRIPT** command lets you execute the **TwinCAD** commands from a given ASCII file. **TwinCAD** will pop up the file window for you to select the script file.

Once a script file is loaded, commands are read in from the beginning to the end of the file. Every character read in from the script file is treated as if it were just typed in from the keyboard. Be careful with the spaces created when preparing a script file.

Should a command error happen during the execution of a command script, the script file will be terminated. You may also abort the script file execution by typing **Ctrl/C** from the keyboard.

An interrupted script file may be resumed by the **RESUME** command. You may use **RSCRIPT** command to restart the script file execution from its beginning, immediately after it is interrupted or terminated.

**Procedure:**

To execute a command script, type as below:

@CMD: **SCRIPT** (*return*)

Script file: (*Operate on file window*)

**Example:**

# SELECT

## General Objects Selection Command

---

### Purpose:

The **SELECT** command is used to mark selected state upon specific objects.

### Description:

The **SELECT** command lets you enter the object selection operation to impose selection states upon specific objects for further processing. The selection operation will be invoked automatically later on by other commands when such operation is required.

When the object selection operation is activated, either by an explicit **SELECT** command or by the request from other commands, **TwinCAD** will prompt

Select Objects (+):

and ask you to select objects. The '+' sign in the prompt represents the active manipulating mode on the object selection state. There are three manipulating modes provided to manipulate the selection state of objects, namely

- **Add mode** - denoted by a '+' sign in the prompt, specifying that all the objects you picked will be imposed with the selected state. This is the default mode.
- **Remove mode** - denoted by a '-' sign in the prompt, specifying that all the objects you picked will be excluded from the selected state, i.e., the selected state will be removed from these objects.
- **Toggle Mode** - denoted by a 'T' character in the prompt, specifying that all the objects you picked will be changed in their states of selection. If the picked object was in selected state, then its selected state will be removed. If the picked object was not in selected state, it will be in.

You may use the following methods or sub-command options to specify the objects for manipulating their states of selection:

- (point)** Direct picking on one object and manipulating its selection state.
- A** **Add**, specifying the Add mode in the selection state manipulation.
- R** **Remove**, specifying the Remove mode in the selection state manipulation.
- T** **Toggle**, specifying the Toggle mode in the selection state manipulation.
- L** **Last**, specifying the last object created.
- P** **Previous**, specifying all the previous-selected objects in the last object selection operation.
- W** **Window**, request to open a window and specifying all the objects within the window. You will be asked to designate the two corners of the window. A dragging box in solid lines will help you doing the job. All the objects that fall completely within the window will be selected by the operation.
- C** **Crossing**, request to open a window and specifying all the objects within or crossing the window boundary. Like **Window** option, You will be asked to designate the two corners of the window.
- BOX** **Box**, request to apply the Crossing or Window mode automatically.

- AU** **Auto**, request to apply the Crossing or Window mode if picking on nothing.
- SI** **Single**, specifying to make only one selection in the object selection. The object selection operation terminates automatically when the selection is made.
- M** **Mask**, request to set up the object selection mask (See **SELMASK** command)
- F** **Fence**, request to apply successive lines to cross objects for the selection. **TwinCAD** will prompt in sequence:

```
Boundary/<First Point>:
```

```
Second point:
```

```
...
```

```
Second point:
```

asking you to designate the first point of the line and then the second point of the line. Objects that cross with the line will be selected. **TwinCAD** will then take the second point of the line as the first point and repeat the prompt for second point of the next crossing line again, until you press space bar to end it.

Instead of giving successive crossing lines to select objects, you may choose to use existing boundary entities to select the crossing objects by the sub-command option "Boundary", which is provided only at the first prompt. See latter paragraph for this mode of operation.

- CP** **Crossing Polygon**, request to apply an irregular close polygon by successive vertex points to select objects that are crossed by the polygon. **TwinCAD** will prompt in sequence:

```
Boundary/<First Point>:
```

```
To point:
```

```
...
```

```
To point:
```

asking you to designate the first vertex point of the polygon and then the next point of it. **TwinCAD** will repeat the prompt for the next point until you press space bar or the sub-command option "Close" to close the polygon. All the objects that are totally within or crossing with this polygon will be selected. During specifying the successive lines, you may enter the sub-command option "U" to undo the last line.

A sub-command option "Boundary" is provided only at the first prompt for you to enter the Boundary mode operation, which allows you to choose existing boundary entities for the crossing purpose. See latter paragraph for this mode of operation.

- WP** **Windowing Polygon**, request to apply a irregular close polygon by successive vertex points to select objects that fall completely within it. It works like the **CP**, excepts that objects must be totally within the polygon or within the boundary entity (under boudnary mode operation) to be selected. See also the description for **CP** option.

- ALL** **All**

selection mask (See **SELMASK** command operation). After that, use the "**ALL**" option to select all objects from the drawing. The mask will then take effect, and only circles on layer 0 are manipulated on their selection states.

Objects which are in selected state will be highlighted in display. However, once the command that invokes the object selection exits, the selected state of that object will be removed and changed to previous selected state. This includes the **SELECT** command itself when it is invoked explicitly. So, all the highlighted objects during the **SELECT** command will be in normal display again after the object selection is over. Nevertheless, the objects which were in previous selected state can be re-selected automatically by the sub-command option "**P**" when the object selection operation is activated again.

Note that the order of selection on the objects does not imply the order of process by the commands which require them. Objects are processed in their order of appearances in the drawing database.

## Boundary Mode Operation

If you choose the Fence, Crossing Polygon or Windowing Polygon methods to select objects, you may choose to select objects by existing boundary entities. To do this, you have to enter further the sub-command option "**Boundary**" to the first prompt that supplied by these methods.

**TwinCAD** will prompt:

Inclusive/eXclusive/<Select Boundary Entity (I/+)>:

asking you to pick up a boundary entity. The valid type of the boundary entity depends on the current selection method. If you are using polygon method (CP or WP), then only Circle and Close Polyline can be picked up and used for the purpose. Otherwise, for Fence method, you may pick Line, Arc, Ellipse, and Open Polyline, in addition to Circle and Close Polyline.

Once a valid boundary entity is picked, **TwinCAD** will apply the specified selection method using the boundary entity immediately, and then repeat the prompt again for the next boundary entity, until you press the space bar to end it.

The following sub-command options are provided:

- A**      **Add**, specifying the Add mode in the selection state manipulation.
- R**      **Remove**, specifying the Remove mode in the selection state manipulation.
- T**      **Toggle**, specifying the Toggle mode in the selection state manipulation.
- I**      **Inclusive**, specifying to include the picked boundary entity in the selection.
- X**      **Exclusive**, specifying to exclude the picked boundary entity from the selection.

## Special Notes

The **AUTO** mode is now the default mode when **SELECT** is entered for multiple objects selection.

If the Selection Mask is enabled, then the color mask and the linetype mask setup by the system variable **COLORMASK** and **LTYPEMASK**, respectively, will be effective in the object selection. You may issue '**CMASK** and '**LTMASK** commands to setup these two masks.

If the bit 0 of the system variable **ENTORDER** is set to 1 prior the object selection operation, **TwinCAD** will create a separate Selection Order List from the object selection operation, which will affect the result returned by the TCL function **getesel()**.

However, it DOES NOT imply that all the command operations will use this order list to process the selected objects. It is still command dependent. Currently, only **DXFOUT** command will use the order list to export selected objects, if it is available.

**Procedure:**

To select objects in advance for the next command, type as below:

@CMD: **SELECT** *(return)*

Select Object (+): *(pick objects or enter options)*

**Example:**

**'SELMASK**

## Set Object Selection Masks Command

---

**Purpose:**

The **SELMASK** command is used to set up the object selection masks used in **SELECT** command.

**Description:**

The **SELMASK** command lets you set up the object selection masks used in the object selection operation (See **SELECT** command). It pops up an Object Selection Mask Dialogue Window for you to set up the masks with window operation. This operation is also invoked by the sub-command option "**M**" in the **SELECT** command.

Currently, there are two selection masks provided for the object selection, namely:

- **Type Mask** - Mask an object by its data type. To set up this mask, move the cursor to pick the box in front of the object type name item. An object type is selectable only if its corresponding item box is marked. The marking of a box is toggled each time you pick it.
- **Layer Mask** - Mask an object by its layer. To set up this mask, also move the cursor to pick the box in front of the name of each layer. You may need to scroll the layer name slide window to access all the layers, if the number of layers is larger than the one which the slide window can hold. Again, a layer is selectable only if its corresponding item box is marked.

Note that the masking function will be in effect only when it is enabled and under the object selection operations. It is different from the object snap function. To enable or disable the masking function, pick up the corresponding box from the window. You may type **<Ctrl/Y>** to toggle the status of masking function when you are picking the objects.

To exit the operation window, press the **<ESC>** key or the rightmost mouse button.



**'SETDIM****Set or Change Dimension Local Properties****Purpose:**

The **SETDIM** command is used to change the local properties of selected dimensions.

**Description:**

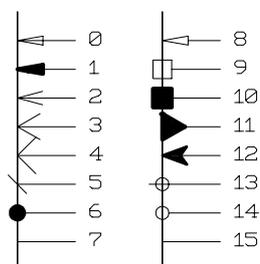
The **SETDIM** command lets you change the local properties of selected dimensions. All the dimensions are created with default properties, with some specified by the dimension variables. These properties, if locally controlled, can be changed individually. Currently supported properties of a dimension which can be changed by this command are:

- **Arrow Type** -- The type of arrow used in the dimensioning. Each arrow of the dimensioning can be locally changed. Currently supported arrow types are from 0 to 15. See dimension variable **DIMATYPE** in **DIMVAR** command.
- **Suppression of Extension Line** -- Each of the extension lines of a linear or angular dimensioning can be suppressed locally.
- **Dimension Text Position** -- The position of the dimension text can be moved to a new location while the rest of the dimensioning remain unchanged. You may also change the text content under this option.
- **Dimension Scale Factor** -- The local dimension scale factor can be changed to a new value which will in turn change the individual size of dimension text, arrow size and so forth.
- **Reset Default Text** -- Dimension text can be reset to default text under current setting of dimension variables. The default text will be generated in association with the true measurement value.

Enter the **SETDIM** command, and **TwinCAD** will prompt:

Extension/Default-text/Scale/Text-position/<Set Arrow type>:

The default operation, if you press the space bar, is to change the arrow type of dimensions. In fact, you may directly enter a value to specify the arrow type to change for the default operation.



For other operations, you should enter the respective sub-command options as described in the sub-sections below.

## Change Arrow Type

If you are going to change the arrow type of selected dimensions, you should enter a null return or a direct arrow type value to the first prompt after issuing the **SETDIM** command. **TwinCAD** will then prompt

<Select dimension by its arrow/arrow type (n)>:

continuously until you type <Ctrl/C> or enter null return again to exit the operation. You may change the arrow type of a dimension by specifying the desired number of arrow type to the prompt, which is also displayed in the prompt, and then picking at the arrow of the dimension to change. The object snap **ENDpoint** is automatically assumed when you are picking on the dimensions.

Note that if a dimension containing two arrows (e.g., linear dimensions) is picked (except the Diametric dimension), the nearer one to the picked point is changed. Note also that leader line may have at most two arrow pointers, though initially only one is enabled.

To effectively remove an arrow pointer, set the arrow type as number 15. See **DIMATYPE** variable in **DIMVAR** command for the supported arrow type.

## Suppression of Extension Line

Each of the extension lines of a linear dimension or an angular dimension may be suppressed or enabled individually. You may enter the sub-command option **"E"** after issuing the **SETDIM** command to request to change the suppression state of extension line of dimensions. The system will prompt:

Set dimension extension line -- OFF/ON/<Select dimensions (On/Off)>:

continuously until you type <Ctrl/C> or enter null return again to exit the operation. The current state of extension line to change to is displayed in the prompt. If it is "OFF", it means the extension line of the dimension you pick will be suppressed, and if it is "ON", the extension line will be generated. You may enter the sub-command option "ON" or "OFF" to switch to the desired state.

Again, you pick up the dimensions of which the states of extension line are to be changed. As each dimension has two extension lines, no matter which extension line you want to suppress, it is the nearer one to the picked point that will be changed.

## Special Notes

You may suppress the dimension line of a Diameter dimension entity using this sub-command option.

If the dimension text is placed inside of the diameter and clips the dimension line into two parts, you may suppress either part of the dimension line.

If the dimension text dose not clip the dimension line, the suppression will truncate the corresponding half of the dimension line and make it passing through the center of the diameter with a minimum extra length equal to twice of the arrow size or a length that extends line out to cover the span of the text.

## Move Dimension Text

The dimension text position may be changed to a new location (though the angle direction can not be changed) by entering the sub-command option **"T"** after issuing the **SETDIM** command. **TwinCAD** will then prompt

Which Dimension:

asking you to select the dimension of which the text is to be moved. Once you have picked up the dimension with text, **TwinCAD** will then prompt

Default-text/Change-text/Angle/<Set dimension text position>:

asking you to designate a point where the text will be moved to. You will be able to drag the dimension text when you are moving the cursor around.

Additional sub-command options "**D**", "**C**" and "**A**" are provided for your convenience. You may enter these sub-command options to reset the dimension text to default text, to change the dimension text manually, or to change the text orientation, respectively.

**TwinCAD** will continue the prompt

Which Dimension:

for the next dimension text to move, until you type a <Ctrl/C> or enter a null return to the prompt to exit the operation.

Note: The sub-command option "**A**" is used to specify the angle of a selected dimension text. Fixing dimension text angle to a specific direction is a new feature in **TwinCAD** and is not backward compatible. The earlier version before V2.0 supports only 'floating' dimension text angle. It will not recognize the fixed dimension text angle, and will determine the dimension text angle based on the dimension rules.

Note: An additional option "**Oblique**" will be added without an explicit keyword in the prompt, if the selected dimension has an oblique feature. This sub-command option is used to toggle the oblique effect on the dimension text and its arrows.

## Change Scale Factor of Selected Dimensions

The dimension scale factor controls the relative size of dimension text, arrow pointers and other length regarding the generation of the dimensions. It is a local parameter saved with each dimension. That means, each dimension may have its own dimension scale factor which could result in different dimension size from others.

To change the scale factor of existing dimensions, you may enter the sub-command option "**S**" after issuing the **SETDIM** command. **TwinCAD** will enter the object selection operation and prompt:

Select Object (+):

for you to select the dimensions that will be changed with new scale factor. The selection mask will be set automatically such that only dimension entities can be selected. If there is any dimension entity being selected, after the object selection operation, **TwinCAD** will prompt:

Enter new dimension scale value (*val*):

asking you to specify the new scale factor for the selected dimensions. The scale factor ranges from 0 to 255.99 with an effective resolution of 1/256. A scale factor of 0 will disable the scaling effect. Negative values are acceptable and will be converted to positive values. Any value out of the specified range is also accepted, not the value itself, but the remainder after divided by 256.

The command exits automatically when the job is done.

Special Note: You may enter the scale factor in the form as "@val" to specify a scale factor relative to the current scale factor of the selected dimensions. For example, entering "@2" will

scale up each of the dimension scale by 2. This should be helpful in determining the actual dimension scale value required for an existing dimension.

## Reset Default Text

The default dimension text for a dimension is generated in association with its dimension measurement under the current setting of dimension variables. It is generated when the dimension is created. The operator may change this dimension text to any text string during the creation or modification of it. And thereafter, the dimension text for that dimension is fixed, until it is modified again explicitly. Though it is still associated with the dimension measurement.

The dimension measurement of a dimension may be changed by drawing editing command such as **SCALE**, and the content of dimension variables may also be changed for new requirement. But these changes will not result in the change of the dimension text of existing dimensions immediately, since it may not be desirable.

However, if you want to change these dimension texts to the default, you may enter the sub-command option "**D**" after issuing the **SETDIM** command. **TwinCAD** will enter the object selection operation and prompt

Select Object (+):

for you to select those dimensions of which the dimension texts are to be reset to the default. The selection mask will be set automatically such that only dimension entities may be selected.

The command will exit when the job is done. Note that you may also use **CHANGE** command to change the dimension text to the default text in a dimension entity. But if the dimension text is the only part to be changed, using **SETDIM** will be much easier.

## Procedure:

- **To change the arrow type of some particular dimensions**

@CMD: **SETDIM** *(return)*

@CMD: SETDIM Extension/<Arrow type>: *(return)*

<Select dimension by its arrow/arrow type (n)>: **tn** *(return)*

<Select dimension by its arrow/arrow type (tn)>: *(pick objects)*

...

- **To change the states of extension lines of some particular dimensions**

@CMD: **SETDIM** *(return)*

@CMD: SETDIM Extension/<Arrow type>: **E** *(return)*

Set dimension extension line -- OFF/ON/<Select dimensions (Off)>: *(pick objects)*

...

Set dimension extension line -- OFF/ON/<Select dimensions (Off)>: **ON** *(return)*

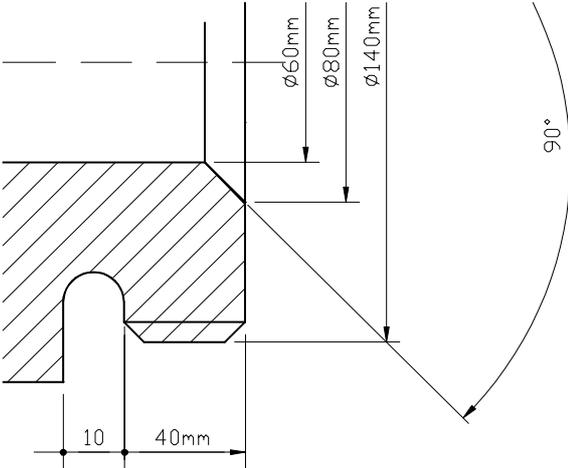
Set dimension extension line -- OFF/ON/<Select dimensions (On)>: *(pick objects)*

...

## Example:

Examples of dimension effects done by SETDIM.

**VDIM** with extension lines suppressed on one side, and one arrow pointer changed to type 6.



# SETWIDTH

## Set wide line property of elementary entities.

---

### Purpose:

The **SETWIDTH** command is used to specify/modify the wide line property of selected elementary entities, such as Lines, Arcs, Circles, Ellipses and Polylines.

### Description:

The **SETWIDTH** command lets you change the wide line property of selected entities.

It will prompt:

Multiple/<Select entity for new line width (nnn)>:

asking you to select the entity to set with the new line width, of which the value is also displayed in the prompt. You may directly enter a value to change the new line width. **SETWIDTH** will repeat the prompt until you answer it with a null return by pressing the space bar or the enter key.

Instead of picking at individual entity for the new line width assignment, you may enter the sub-command option "Multiple" to specify to select multiple entities at one time. **SETWIDTH** will let you select multiple entities through the Object Selection Operation. See **SELECT** command for details of the operation.

An implicit sub-command option "Undo" is also supported for you to restore the last change of entities on wide line property.

To disable the wide line property of an entity, enter a zero value to the prompt for the new line width value, and pick the entity. Entities that can have wide line property are Lines, Arcs, Circles, Ellipses and Polylines.

### The Wide Line Property

The wide line property is a special property that applies only to elementary geometry entities, such as Lines, Arcs, Circles, Ellipse and Polylines. It specifies that the generation of such entities will be using wide line segments that are stroked with a specific pen width value.

Apart from the line width concept brought by **AutoCAD** in its support of the Polyline entities, the wide line property is not a geometry property of the entities, but a generation property, the same as the Color property, of the entities. It affects only the generation of the entities in the output device and is intended for line drawing representation. However, the property value is also called the line width of the entities.

**TwinCAD** does not allow an entity with an extrusion thickness to be generated using wide lines. This is because an entity with an extrusion thickness is, in geometry sense, a wall surface. So, if an entities is set with an effective wide line property (with a non-zero line width value), it's extrusion thickness will be ignored and effectively become zero.

A few notes about the wide line property are given below:

- The width effect generated by an entity with a wide line property is only a generation property of the entity, not part of its geometry. You can not expect to **EXPLODE** it into other elementary entities, nor to snap a point on the edge of the width effect.

- If an entity is marked to have a wide line property, its extrusion thickness will be invalid (zero thickness is assumed).
- The pattern linetype generation for such a wide line is effective. However, it will be done in precise calculation; the fast linetype generation feature (controlled by **FASTLTYPE** system variable) will never be effective to such entity. Also, the dot specification of a linetype will be generated as a real 'dot' under the line width specification.
- The arc or elliptic arc segment will be segmented into successive line segments to generate the width effect. The fineness is thus controlled by the system variable **ARCSEGANG**.
- Beveling between successive wide lines is supported on polyline entity with this wide line property. The content of the system variable **BEVANGLE** controls the way to create the beveling.
- The generation of the solid-fill effect on the area created by the line width effect is also controlled by the system variable **FILLMODE**.
- Since this feature is supported only after version 3.0 (inclusive), drawing entities created with this wide line property will be generated as entities with a non-zero extrusion thickness in earlier version when the drawing is loaded.

**Procedure:**

@CMD: **SETWIDTH** (*return*)

Multiple/<Select entity for new line width (0.)>:

...

**'SHOWDIR**

## Entity Direction Display Control Command

---

**Purpose:**

The **SHOWDIR** command is used to control the display of entity direction.

**Description:**

The **SHOWDIR** command lets you control the function to display entity direction. When the entity direction display function is ON, the direction of all LINES, ARCs and POLYLINES will be indicated by arrow pointers at their ends. These arrow pointers are not part of these entities, but are added only for showing the directions of them.

When you issue this command, if the current entity direction display function is ON, it will be turned OFF. If it is OFF, it will be turned ON and you will have to enter the size of arrow pointer. No prompt will appear for the entry of arrow size. You may simply type in the value after the **SHOWDIR** command.

Note that this function is effective for plan view only. In 3D view, the direction arrow pointers will not be added in current release. The drawing will be regenerated after this command is issued.

**Special Note**

This command is provided mainly for diagnostic purpose. The direction of a LINE, ARC or even POLYLINE should be irrelevant to the drafting activities. When you draw a box, it is a box. When it is further edited or plotted out, it does not matter whether the box was drawn counter-clockwise or clockwise.

Some of the editing commands may modify the direction of entities, and the **UNDO** command may not restore them back. These are acceptable to a CAD system since the direction of entities are not concerned for the drafting activity. THERE IS NO LIABILITY FOR THE DEVELOPER TO KEEP THE DIRECTION OF ENTITIES UNCHANGED WHEN THEY ARE EDITED.

**Procedure:**

To turn on the entity direction display function with arrow pointer of 1 unit in size, type as below:

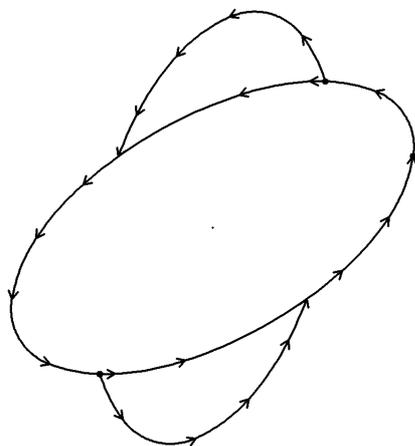
@CMD: **SHOWDIR** *(space)* 1. *(return)*

To turn it off:

@CMD: **SHOWDIR** *(return)*

**Example:**

The figure shows an example display of entity directions. Note that the DOT arrow is used to indicate the start of a polyline.



# SLEADER

## Create a Leader Spline Command

---

### Purpose:

The **SLEADER** command is used to create leader spline.

### Description:

The **SLEADER** command lets you create a leader spline. You create such a leader by designating the start point of the leader first, and then, drag the leader line segment out by designating more points, in the same way as you do at the **LEADER** command.

In fact, **SLEADER** command is actually **LEADER** command with the Spline Option being enabled, such that spline segment will be used to connect these node points instead of using the line segments, after the leader entity is created. See **LEADER** command for more details.

### Procedure:

Type **SLEADER** to the command prompt, and follow the procedure as below:

@CMD: **SLEADER**    *(return)*  
Indicate pointer location:    *(point)*  
Lead line to:    *(point)*  
Undo/<Lead line to>:    *(point)*

### Example:

**'SNAP**

## Change Snap Resolution and Status Command

---

**Purpose:**

The **SNAP** command is used to control the snap resolution and function status.

**Description:**

The **SNAP** command lets you change the snap resolution, axis direction and base coordinate, or turn the cursor snap function ON or OFF. The snap resolution is the spacing between virtual grid points with which the cursor must align. The snap axis direction defines the direction of the grid points in distribution and the snap base coordinate defines a point that must align with one of the grid points. If the cursor snap function is on, the cursor movement will be constrained to align with these points, and so is the data point designated by the cursor.

When you enter the **SNAP** command, **TwinCAD** will prompt

```
Snap Control -- ON/OFF/Aspect/Rotate/Standard/Isometric/  
/ <spacing:nn/base:(xx,yy)>:
```

In response to the prompt, you may input the following options:

*(point)* Point designation, specifying the base coordinate of the snap function.

*(value)* Single value, specifying the spacing of the snap function on both axes.

**ON** **Snap On**, request to turn on the snap function.

**OFF** **Snap Off**, request to turn off the snap function.

**A** **Aspect**, request to set different snap spacing on both axes. You will be asked to input the spacing for the X and Y axis respectively, as **TwinCAD** prompts in sequence:

```
X-direction Spacing <xs>:
```

```
Y-direction Spacing <ys>:
```

**R** **Rotate**, request to set the snap base and the axis directions. You will be asked to designate the base coordinate and the direction of the snap axes (X and Y) respectively, as **TwinCAD** prompts in sequence:

```
Base Point <xx,yy>:
```

```
X-direction Angle <0°>:
```

```
Y-direction Angle <90°>:
```

**S** **Standard**, request to reset the snap axis directions to the standard mode (X-axis in 0°, and Y-axis in 90°).

**I** **Isometric**, request to set the snap axis directions to conform with the Isometric plane (Left, index 0). See also **ISOPLANE** command.

You may use the **DMODE** command to change the snap function with a dialogue window, or use the **<Ctrl/B>** command to toggle the cursor snap function ON or OFF.

**Procedure:**

- **To set snap resolution to 5., type as below:**

```
@CMD: SNAP (return)
```

Snap Control -- ON/OFF/Aspect/Rotate/Standard/Isometric/  
/<spacing:10/base:(0,0)>: **5** (*return*)

- **To turn cursor snap function ON, type as below:**

@CMD: **SNAP** (*return*)

Snap Control -- ON/OFF/Aspect/Rotate/Standard/Isometric/  
/<spacing:5/base:(0,0)>: **ON** (*return*)

**Example:**

# SPLINE

## Spline Curve Generation Command

---

### Purpose:

The **SPLINE** command is used to generate a simple spline from a given polyline.

### Description:

The **SPLINE** command lets you quickly generate a spline from a given polyline. It employs successive **SARC** commands to transform the specified polyline into a smooth curve with continuous arcs, which forms another new polyline.

The transformation takes each successive line segment along the polyline as the tangent to the spline at the point which could be

- the start point of a *starting line segment*,
- the end point of an *ending line segment*, and
- the middle point of a line segment which is between the starting and ending line segment.

A *starting line segment* is one whose start point is adjacent to no other line segments. Likewise, an *ending line segment* is one whose end point is adjacent to no other line segments either. A single line segment can either be a starting line segment or an ending line segment. All arc segments and single line segments are copied during the transformation.

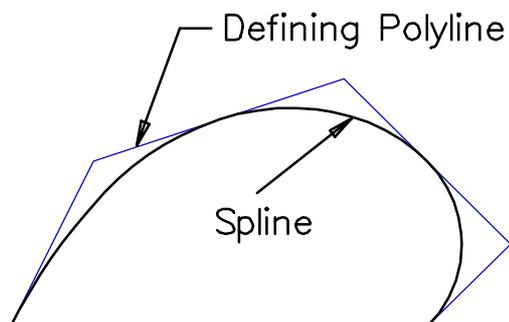
When you issue the **SPLINE** command, you will be asked to select the polyline to transform. After the transformation, a new polyline will be created and you will be asked whether to keep the original polyline or not. If you choose not to, the original one will be erased.

### Procedure:

To transform a polyline into a spline, follow the procedure below:

```
@CMD: SPLINE (return)
Select polyline: (pick one)
Done! -- New polyline with nnn entities segments created.
Delete old polyline? <N>: (Y or N)
```

### Example:



# STRETCH

## Stretch Objects Command

---

### Purpose:

The **STRETCH** command is used to move a part of a drawing while maintaining the connection between the moved part and the rest of the drawing.

### Description:

The **STRETCH** command lets you move a selected part of a drawing while maintaining a link between the moved part and the rest of the drawing. For example, a line is stretched with one end moved to some new place, while the other end stays where it was. The moved part is still connected to the rest of the drawing through the aforementioned line.

Such connection is valid only for Lines, Arcs and Polylines.

To stretch a part of a drawing, you are required to select the objects to move first. See **SELECT** command for object selection. During the object selection, at least one window-style selection must be used. The last window used will be the stretching window moved by the **STRETCH** command.

If you do not apply any window operation during the object selection, i.e., there is no default stretching window setup, the **STRETCH** command will issue an error message and then ask you to specify the stretching window explicitly, as the prompt in below sequence:

You must select a window to stretch.

First corner:

Second corner:

You are then asked to move the stretching window by specifying a base point and a new point as you do for the **MOVE** command. A sub-command option "**Window**" is provided to both the command prompts for the base point and for the second point. You may enter this option to re-specify the stretching window if necessary.

Once the displacement of the stretching window is determined, all objects selected inside the stretching window will be moved to the new place. Those selected objects crossing the window will be moved while maintaining their connections with the drawing outside the window.

### Stretching Linear Dimension Entity

Linear dimension entities can be stretched by their dimension points. The dimension being stretched will be set with its default text (to be associative with respect to change of its dimension points). Details of rules are stated below:

1. To stretch a linear dimension entity, one of its dimension point must be crossed over by the stretching window. If both dimension points are crossed over, it will be moved totally by the stretching displacement only. If neither one is crossed over, nothing will happen. It is the dimension points that determine the dimensioning. You can not use the **STRETCH** command to stretch only the extension lines.
2. Once a dimension point is altered by the stretching displacement, a re-dimensioning will automatically happen and the dimension entity will be modified such that its original characteristic will be preserved while its dimension value is changed to reflect the change

of the dimension points. The regeneration of the dimension text will assume the effect of the current setting of the dimension variables.

### **Stretching Ordinate Dimension Entity**

Ordinate dimension entities created by **XDIM/YDIM** commands can be stretched by the following rules:

1. If both the dimension point and the extension point are inside the stretching window, the dimension is moved totally.
2. If only the extension point is inside the window, the extension point as well as the text

# STYLE

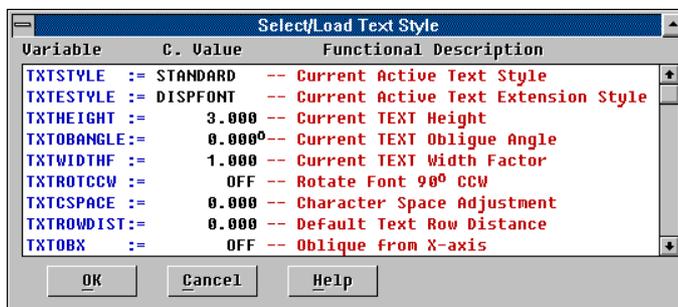
## Set Current Text Style Command

### Purpose:

The **STYLE** command is used to set up the default parameters of text style for the subsequent **TEXT** creations.

### Description:

The **STYLE** command lets you specify the current default parameters of the text style for the subsequent **TEXT** creations. It will pop up a variable-access window for you to change these parameters. You just pick up the item you desire from the window, and change it with further operations.



It is recommended to use **DDSTYLE** command instead of this command.

The following describes each item from the style setup dialogue window:

- TXTSTYLE** Main text style name, specifies the main text style name in use. If you pick up this item, **TwinCAD** will pop up another style name window for you to pick up the one you need, or to load one from the disk. The main text style is used for text font in single code, like ASCII-coded English.
- TXTESTYLE** Extension text style name, specifies the extension text style name in use. If you pick up this item, **TwinCAD** will pop up another extension style name window for you to pick up the one you need, or to load one from the disk. The extension style is used for text font in two-byte codes, like Chinese or Korean, or in variable-length codes such as Thai. These text styles are optional packages to the users.
- TXTHEIGHT** Real value, specifies the size of text font in height. If this value is negative, the text will be mirrored with respect to the base writing line.
- TXTOBANGLE** Real value, specifies the oblique angle of the text font to generate, in units of degrees. It is normally 0.
- TXTWIDTHF** Real value, specifies the width factor of the text font to generate. It is normally 1. If this value is negative, the text will be mirrored to the opposite of the writing direction.
- TXTROTCCW** On/Off state selection, specifies to rotate the text font by 90° Counter-ClockWise or not. It is normally OFF. The capability to rotate the font by

90° CCW is useful in writing the Chinese/Korean texts or the likes in vertical manner.

**TXTOBX** On/Off state selection, specifies that the new Text entity in creation will be obliqued from the baseline if the oblique angle is not zero.

**TXTCSPACE** Real value, specifies the character space adjustment for new created TEXT entities. A positive value specifies absolute space (additional to the original one defined by the font), negative value specified relative space ratio over the text height. A zero value will disable this function.

**TXTROWDIST** Real value, controls the default text row distance by the following rules:

**1. If it is zero**, which is the default, the default text row distance will be taken from the text font via the linefeed code. This provides the compatible mode with the earlier version. However, should the definition of the linefeed code in the text font file be missing, a text row distance of 1.5 times of the text height will be assumed.

**2. If it is greater than zero**, the value will be directly taken as the absolute row distance for subsequent text generation.

**3. If it is negative** (less than zero), its absolute value will be taken to specify the ratio of the text row distance over the text height. That is, the text row distance will be the text height multiplied by the absolute value of this variable.

Note that the default text row distance will be scaled down and up automatically by the shift-in/out functions ('{' and '}').

It is recommended to set this value to a negative value, say -1.5 or -2.0, since some of the text font files may not have a proper definition for the linefeed codes.

## Loading a text style from disk

If you want to load a text style from disk, follow the procedure described below:

1. Select the type of text style to load from the pop-up window under **STYLE** command.
2. Pick the <NewStyle> item and enter the style name (as you like), then press return.
3. **TwinCAD** will pop up a file window for you to select the file that contains the specific text font. Note that the font file with an extension of "**SHZ**" is for main text style, and "**SHE**", for the extension style.
4. If no error occurs, the text style will become the current text style. A report of style loading will be given at the command line as the examples given below:

```
STANDARD text style loaded from D:\TCAD\SUPPORT\STANDARD.SHZ
COB (CountryBlueprint) text style loaded from C:\PFB\COBT____.PFB [FILL=OK]
```

Note that the message "**[FILL=OK]**" indicates that the text font can be solid fill. You may use FILL command to generate solid fill effect over the text entities using this text font.

Possible errors in loading a text font are as below:

```
Unauthorized use of optional text style xxxxxxxx
```

You are trying to load a protected text font style which is not yet registered to **TCAM Development House**. For optional text style of authorized version, ask for your local dealer.

# 'SWITCH

## Switching for Different System Setup Command

---

### Purpose:

The **SWITCH** command is used to switch the current system setup to another one defined by other system initial files (**INI**) for the DOS version, and to switch between the default setup and alternate setup for the Windows version.

### Description:

#### Windows Specific

The **SWITCH** command lets you switch **TwinCAD** between the Default language setup and Alternate language setup, which define the TGF, Menu, Help file and variable information files in use and are specified from the system initial file.

The **SWITCH** command in **TwinCAD** for Windows will not ask for the selection of a different system initial file as it does in the DOS version. When you enter the **SWITCH** command, **TwinCAD** will simply toggle the current language setup between the Default and the Alternate one. See **The System Initial File** for more informations on these setups.

#### DOS Specific

The **SWITCH** command lets you re-initialize **TwinCAD** for a different operation setup by re-loading in a different initial file (**INI**) from the disk. The initial file should contain proper information about the screen layout, graphic runtime driver and language text used, and other related parameters setups. See Installation Guide for details of the INItial files.

**TwinCAD** initializes itself with the information from the default INItial file or the one specified explicitly from the command line, as soon as it starts the execution. You may re-initialize it by entering the **SWITCH** command without saving the work and leaving **TwinCAD**.

When you enter the **SWITCH** command, you will be asked to select from the pop-up file window the specific INItial file to load in. After that, the graphic display will be cleared and changed to text mode (in single screen system), and the message

```
[TwinCAD]: Loading Initialization File: INI-filename
```

will be displayed when **TwinCAD** is loading the initial file. Should there be any error, the messages

```
[TwinCAD]: Initial file contains errors
Switching aborted!  -- Press any key to continue.
```

will be displayed and the original INItial file will be re-loaded in again. If the newly loaded initial file contains different graphic runtime driver, **TwinCAD** will prompt

```
[TwinCAD]: Initial file contain different Graphic Runtime:
Original: Old-driver-name
New One: New-driver-name
Select A)bort, L)oad it, I)gnore it :
```

asking you to select from the possible actions. Be aware of the fact that **TwinCAD** won't work for wrong graphic driver setup. You can't expect a monochrome monitor to be driven by a

VGA graphic driver. However, an EGA graphic driver may possibly drive a VGA monitor. So, make sure that your selection is correct when this situation happens.

The re-initialization will cause **TwinCAD** to

- re-load the text generation file (TGF), so that message texts of different language version can be switched in.
- re-set up the display and screen layout
- re-load the menu file
- and refresh the drawing screen.

You may continue your work after the switching is done, as if nothing happened. You can't undo the **SWITCH** command. While you may switch it back by issuing the **SWITCH** command again.

### Procedure:

To switch to different system setup, type as below:

```
@CMD: SWITCH (return)  
(Select Initial file from file window)  
(Switching taking place...)
```

### Example:

# SYMBOL

## Insert External Symbol Command

---

### Purpose:

The **SYMBOL** command is used to insert an external symbol reference into the drawing.

### Description:

The **SYMBOL** command lets you insert an external symbol reference at a particular position in the drawing with specified scale factor, direction of alignment, and possibly attribute tag value entries.

When you enter the **SYMBOL** command, **TwinCAD** will firstly ask you to enter the name of the symbol to insert and where to load it by the following prompt:

Which Symbol (or ?) <*symname*>:

where the *symname* is the Symbol name of last reference and is taken as default if you reply to it with a null return.

You specify the Symbol name by typing it from the keyboard in the format as below:

`{?}{symname{={symlibname}}}(spacebar or return)`

where the portions enclosed with braces are optional. You can see all of them are optional except the last one which terminates the input string. The meaning of each portion is described below:

- **?** - An optional question mark, given immediately after the prompt, tells **TwinCAD** to pop up a symbol selection window to select the symbol from the current active symbol library. All the defined symbol names from the library are in the slide window, so that you don't need to memorize nor to type the name you desire, but to select it from the window. All the rest of the characters in the input string to the optional equal sign are ignored. If there is no symbol library active in the drawing and there is no symbol library specified in the rest of the string, **TwinCAD** will pop up the library selection window first to select the active library.
- **symname** - Character name string, specifying the name of the symbol to insert. If the specified symbol is not yet loaded, the current active library will be searched for the loading of it. If it can not be found in the active library, **TwinCAD** will pop up the library selection window for you to specify where to find it.
- **=** - An optional equal sign, immediately after the symbol name, tells **TwinCAD** to load the symbol from the symbol library specified by the string following it. If there isn't any character string, which specifies the filename of the symbol library explicitly, following this equal sign, **TwinCAD** will pop up the symbol library selection window for you to indicate where to load the symbol. The symbol library thus selected will become the current active library.
- **Symlibname** - An optional file name of symbol library, immediately after the equal sign, specifies where to load the symbol explicitly. The extension of this symbol library file is always **"SMB"**. You don't need to type out this extension. If this symbol library can not be found, the symbol library selection window will again be popped up.

After the symbol is well specified, **TwinCAD** will load it in and ask you to designate the base point of insertion while dragging the image of the symbol with the following prompt:

Symbol insert base point:

Once the insertion point is given, you will be asked to determine the size of the symbol by answering to the prompt:

Size/<Symbol size by scale factor (1)>:

with a scale factor. You may designate a reference point directly to determine the symbol's insertion scale and rotation angle, as the dragging image implies when you move around the cursor pointer. The angle direction from the insert point to this designated reference point will be used as the rotation angle of the symbol. And, the distance between these two points will be used as the size of the symbol in terms of its original defining bounding box. Note that, once the scale factor and the rotation angle are determined by the designated point, the symbol is inserted without asking further the oblique angle nor the width factor.

Besides of giving a scaling factor and designating a reference point, You may enter the sub-command option "**S**" to specify to enter the absolute size of the symbol in drawing unit, as it prompts:

Enter height of symbol <nnn>:

The size of a symbol so specified will be the height of the symbol inserted. A proper scale factor will be calculated for the insertion.

The next step is to specify the alignment angle of the symbol, as the prompt below:

Width/Oblique/Symbol alignment angle <nn°>:

You may use the object snap directive **DIR** to align the symbol or directly give the angle value in units of degree. Two additional sub-command options are provided here:

- W**     **Width**, Request to specify the optional width factor of the symbol. The default width factor is 1, which keeps the aspect ratio of the symbol the same as its original definition. You will be asked to change the width of a symbol by entering a new width factor for it. Negative value of this width factor will have mirroring effect.
- O**     **Oblique**, Request to specify the optional oblique angle of the symbol. The default value for this oblique angle is 0, which produces no oblique effect at all. However, you may specify the oblique angle (the same definition as that for a Text) to make the symbol insertion look like being projected in 3-D view.

Finally, if the symbol being referenced contains variable attribute tags for display purpose, **TwinCAD** will open the attribute tag window for you to enter attribute values to each of them.

And thus, the insertion of a symbol is completed.

## Texts In A Symbol

The symbol may contain constant TEXTs in its original definition. However, the names of the text font style used for these TEXTs can not be included in the symbol definition. Therefore, the generation of such constant texts in a symbol must be globally controlled by the system variables. Two of such variables are used to do the job:

- SYMSTYLE**     Specifying the style name of the **English Font** used for the constant text generation in symbols, default to **STANDARD**.
- SYMESTYLE**    Specifying the style name of the **Extended Font** used for the constant text generation in symbols, default to nothing.

Changing these variables will change the text generation of the symbols.

Another possible text generation in symbols is the one for the viewable attribute tags. The font styles used in such text generation are controlled by the current Text Styles used for Texts in creation, not from these two variables, in a local manner. When a symbol is inserted, the style names specified by the current Text Styles will be kept local with the attribute tags, since the attribute tags are local. So, changing these variable or the current Text Style will not change the variable text generation from attribute tags of the symbols.

### Special Note about Text in Symbol

If the Text was inserted without any justification requirement in the original symbol definition (i.e., the default base-left justification was used), the constant text in symbol will follow the rules stated earlier. The generation of such Texts will be globally controlled by the system variable **SYMSTYLE** and **SYMESTYLE**.

However, if the text was inserted with any justification requirement other than the Left (base line) justification, the constant text in symbol will include additional informations (such as oblique angle, width factor, justification, character distance, etc.). Since such texts require justification other than the Left, the starting pen position in writing the texts needs to be recalculated each time they are generated. It is then not practical to have a global control over such texts.

The generation of such Texts are locally determined when the symbol is created. Tags of "**CONST\_TEXT**" will be created to maintain such text informations locally. That is, these texts will be converted to Tags named "**CONST\_TEXT**" at symbol insertion.

### Colors In A Symbol

A symbol may be drawn with more than two colors. However, since only one color is possible with the insertion of the symbol, the symbol must possess its own local colors for the elements that must be drawn in different colors.

This is done by defining the symbol with entities of different colors. If an entity, used in the definition of a symbol, assumes a display color of number 1, the part of the symbol it contributes will possess a local symbol color number 1. If the entity assumes a display color By Layer, then the corresponding part in the symbol will assume the color by the insertion of the symbol (the 'layer' where it stands, in a sense). So are the other possible colors for the symbol's elements.

These local symbol color numbers, from 0 to 15, does not imply the pen color number from 0 to 15. You may assign the local symbol color number 1 to use the pen color number 5, or to assume also the same color of the symbol's insertion, by using the **SCOLOR** command. See **SCOLOR** command for details. The initial default assignments for these local symbol colors are [**By Layer**], which means to assume the same color of the symbol's insertion. The assignments are global to all the symbols.

### Inserting Symbol on PUCS-plane

The **SYMBOL** command is able to recognize the current **PUCS**-plane setup of an Axonometric Projection, and allows you to insert the symbol on the projected plane from the virtual space directly. This means, **TwinCAD** will automatically set the required width factor and oblique angle for the new insertion of symbol, such that it will look correctly in the projection view.

If the insertion angle is given in relative angle (in the format as '@ang'), it will be taken as the angle for the symbol image to be rotated in the virtual 3-D space. The same effect can be

done by rotating the current **PUCS** axes around its Z-axis for the specified angle (in the virtual space) and then insert the symbol under the new **PUCS** plane with a zero angle (relative to the **PUCS** X-axis).

### Special Note

When you are selecting a symbol from the symbol selection window, which contains the names of all the symbol from current active symbol library, you may switch to another symbol library by picking up the item "<<Change Symbol Library>>" from the selection window.

Note that a symbol is not a block instance. Don't expect it to behave like a block instance. For example, you can not explode a symbol like you do to a block instance.

### Procedure:

- To insert a symbol named NUT by a given size of 10mm:  
@CMD: **SYMBOL** (*return*)  
Which Symbol (or ?) <symname>: **NUT**  
Symbol insert base point: (*point*)  
Size/<Symbol size by scale factor (1)>: **S**  
Enter height of symbol <nnn>: **10**  
Width/Oblique/Symbol alignment angle <nn°>:**0**
- To insert a symbol named RESISTOR from a symbol library named ELECPART.SMB:  
@CMD: **SYMBOL** (*return*)  
Which Symbol (or ?) <symname>: **RESISTOR=ELECPART**  
Symbol insert base point: (*point*)  
Size/<Symbol size by scale factor (1)>: (*as wish*)  
Width/Oblique/Symbol alignment angle <nn°>: (*as wish*)  
(*Enter the resister value to attribute tag from the popped up window*)

### Example:

## Create/Update Symbol Libraries Command

---

### Purpose:

The **SYMLIB** command is used to create or update symbol libraries.

### Description:

The **SYMLIB** command lets you create new symbol library or update a symbol library with newly defined symbol. A symbol library is a collection of symbols in the form of a disk file. Such a collection may contain unlimited number of symbols. A symbol is defined by a group of drawing containing lines, arcs, circles, and many other possible elements, and is commonly referenced by its name for showing purpose. You may use **SYMBOL** command to insert a reference of such symbol.

### Create New Symbol Library

To create a new symbol library, enter this **SYMLIB** command, and pick up the item:

<<Create New Symbol Library>>

from the Symbol Library Selection Window popped up thereafter. The system will open a data entry field at the same line for you to enter the usage description text for the symbol library to create. This description text may be given in any language supported. You have to type the Enter key to terminate the text entry. This usage description text is displayed in this symbol library selection window to represent the name of the symbol library in selection.

After the description text entry, a file window will be popped up for you to specify the name of the symbol library to create. Once this is complete, an empty symbol library will be created as required. This newly created symbol library will become the current active symbol library.

**TwinCAD** will continue the operation procedure to define a new symbol and then add it into the newly created symbol library. You may quit the operation by picking at the [-] screen button of the window, or pressing the <ESC> key while in the window operation, or typing Ctrl/C to the command prompt if applicable.

### Update Old Symbol Library

To update an old symbol library, enter the **SYMLIB** command, and pick up the item:

the symbol library to update.

Valid characters are alphanumeric, '-' (minus), '\$' (dollar), '\_' (underscore), '/' (slash), '@' (At sign) and '#' (number) characters. All lower case characters will be converted to upper case.

If you pick up an existing symbol name, you are to redefine the symbol. **TwinCAD** will prompt

```
Symbol 'xxxxxxx' already exists!  
Are you sure to redefine it ? <N>:
```

to report the situation and ask for further confirmation. If your answer is negative, the operation will exit. However, if your answer is positive, the new symbol defined in the next coming procedure will replace the selected one from the symbol library.

## Define A New Symbol

After the completion of the specification of the symbol name and the symbol library, the next step is to define the bodies of the symbol.

Before entering the **SYMLIB** command, you should have already drawn the shape of the symbol to define. Valid entities for the shape of a symbol are **LINEs**, **ARC**s, **CIRCLE**s, **POLYLINE**s, **ATTRIBUTE**'s, **TEXT**s and **REGION**s. The possible solid color fill of these entities are also accepted. Linetypes are generated by decomposition as a series of line or arc segments. The color number assumed by these entities will become the local symbol color number they assume in the symbol, except the one [**By Layer**] which specifies that the color of the entities in the symbol will follow the insertion of the symbol.

The procedure to define a new symbol is about the same as that to define a Block. You will be asked first to designate the insertion base point of the symbol:

```
Insert base point:
```

And then, you will be asked to select those valid objects that comprise of the symbol shape via the Object Selection Operation.

Finally, if your drawing contains some predefined groups of attribute tags (via **TAGVAR** and **TAGS** command) and the content of the system variable **ASKFORTAG** is not zero, then, after the object selection operation, **TwinCAD** will prompt the question:

```
Tags available. Do you want to add tags to the block? <Y>
```

the same as the one for block definition, asking you whether to tag the block of drawing with attribute tags or not. If your answer is positive (by default), **TwinCAD** will pop up the tag group selection window from which you can choose the desired tag group to apply to the symbol automatically.

A tag group is a group of attribute tag definitions. If there is no tag group available in the drawing, then no attribute tags will be included to the newly defined symbol. You can use **TAGVAR** command to define attribute tags (including attribute name, type of tag and prompt), and then use **TAGS** command to create groups of these definitions. Note that only those Viewable attribute tags will be included in the symbol; they are not necessarily Variable. All tags in symbol will assume Variable attribute automatically at symbol insertion. See **TAGS** and **TAGVAR** command reference.

At last, the symbol is defined and stored in the symbol library with the given name. The drawing entities that are used to define the symbol will remain intact after the operation.

The symbol so created will have a type value of 0. **TwinCAD** plans to support other types of symbol in the future release.

## Special Notes

The attribute tag definition in Text entity is also recognized and converted into equivalent Tag entity before the symbol conversion. See "Using TEXT as ATTDEF (Attribute Definition)" section in **BLOCK** command. However, all the non-visible and constant tag will still be ignored in building the symbol with tags.

All Texts in symbols share the same global text style setting. However, only those TEXT entities with effective width factor, character distance setting, oblique angle and rotated CCW property, will be converted into CONST\_TEXT tag element in the symbol. Such CONST\_TEXTs shall maintain its own local text control when the symbol is inserted.

## Procedure:

@CMD: **SYMLIB** (*return*)  
(*Operate on window to select symbol library*)  
(*Operate on window to specify the name of symbol to define*)  
Insert base point: (*point*)  
Select Object (+): (*do so*)  
...

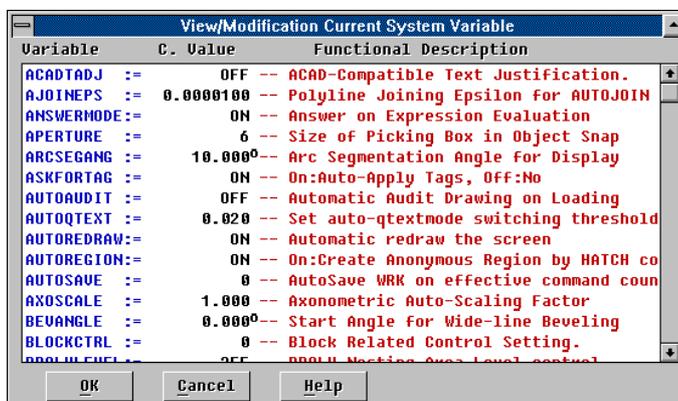
## Example:

**'SYSVAR****View/Modify System Variables Command****Purpose:**

The **SYSVAR** command is used to read/write access the system variables via a dialogue window manner.

**Description:**

The **SYSVAR** command lets you access the system variables in a slide window manner. When you enter the **SYSVAR** command, the system will pop up a slide window for you to view or to change the content of the system variables. To change a variable, just pick it up by the cursor and **TwinCAD** will let you modify its content.



Note that the ordering of these variables from the slide window may not follow any sorting rules. They are so arranged and believed by the developer that the most frequently accessed variables come first. This will be changed in the future release when the number of system variables increases. (Note also that not all the system variables will be listed here, since some are accessed by other commands and some are prepared for TCL programming. The listing should contain only those accessed by **SYSVAR** command).

However, the meaning of each variable from the window is described below in alphabetical order:

**ACADTADJ**

Integer, On-off control, Default = Off. The content of this variable specifies how the text justification calculation will be done for all the texts in generation. If it is ON, the system will use the AutoCAD's method to do the text justification calculation, which uses the actual text stroke extent for the center or right justification control. If it is OFF, which is the default, the system will do the calculation using the text stroke bounding box, in its native way. The reason to provide this additional control is to make the drawing that contains texts with right/center justification control loaded from a DWG file to appear the same as it was in AutoCAD, especially when the texts are drawn to align within some frames. The native text justification calculation provided by the system is different from that provided by the AutoCAD. The system always uses the text bounding box for the justification calculation, while it is the actual text stroke extent used by AutoCAD for the same calculation.

<b>AJOINEPS</b>	Real value, specifies the epsilon value that determines whether two points coincide in the <b>AUTOJOIN</b> and <b>PJOIN</b> command. See <b>AUTOJOIN</b> command.
<b>ANSWERMODE</b>	Integer, On/Off status control, enables or disables the display of the answer of an evaluated expression at the command line. The default is ON. The display of the evaluated answer may also be suppressed if the expression is prefixed with a '#' mark.
<b>APERTURE</b>	Integer, specifies the size of the target box used in object snapping, in unit of pixel. It is accessed also by <b>APERTURE</b> command. See <b>APERTURE</b> command.
<b>ARCSEGANG</b>	Real angle value displayed in units of degree, specifies the increment angle value used to generate a circular segment for the thickness display in a 3D view. Internal default is 10°.
<b>ASKFORTAG</b>	Integer, On-off control, default = ON. This variable specifies whether the system will automatically ask the user to apply tags to the new block definition at <b>BLOCK</b> command or to the symbol in creation in <b>SYMLIB</b> command, when tag groups are available. If it is ON and there are tag groups available in the drawing, the system will ask the user whether to apply tags or not. If it is OFF, the query will be bypassed and no tags will be applied automatically.
<b>AUTOAUDIT</b>	Integer, On/Off status control, enables or disable the auto-audition function. When it is ON, all newly loaded drawing (caused by <b>LOAD/INSERT/NEW</b> ) will be audited automatically. The default is OFF.
<b>AUTOQTEXT</b>	Real value, specifies the threshold control value, which is the minimum ratio of the text size to the height of the current drawing window, used in the auto-switching between quick-text box display and real vector generation of the text entities in display.

When the quick text mode is OFF, all the text entities will be drawn exactly what they are in the drawing window, even if the text is relatively very small in size to recognize. It takes time to generate these text vectors if the drawing contains a lot of texts of fancy styles, especially when the drawing window is zoomed to view the whole large drawing.

To solve this problem, a quick text mode threshold control is provided, which causes the display of a text entity in the graphic window as a single box when the ratio of the text size in height to the height of the drawing window is smaller than this threshold value.

And thus, these small text in the drawing window will be drawn far more quickly than ever as boxes, while you still can read the other texts that are large enough to recognize from the window. To view these small texts, just to zoom the drawing so that they become large enough to display.

To effectively disable this threshold control function, set a zero to this variable. It is effective only when the quick text mode is off.

Note that you may turn the quick text mode ON to make the drawing display faster; however, this will make all the text entities be displayed as boxes. Once you want to read the text, you will have to turn it off and regenerate the drawing again.

- AUTOREDRAW** Integer, controls the auto screen refresh during command operations. Most of the editing commands can recognize the setting of this variable and will not refresh the display list at end of their command operation.
- AUTOREGION** Integer, On-Off control, volatile, default = On. The content of this variable specifies whether the **HATCH** command will create a region entity to be associated with the user-defined hatch pattern given by solid hatching lines. If it is ON, a region entity will be created to cover the hatch area, and the hatching lines specified by the user will become an associated hatching specification with that region. If it is OFF, the **HATCH** command creates a drawing block of the real hatching lines. Note that the **HATCH** command will always create a region entity for the hatching lines generation in an associative manner if a predefined hatch pattern is used.
- AUTOSAVE** Integer, controls the autosave feature. If it is zero (as default), the autosave feature is turned off. Any other values specify the interval of autosaving function to take place (to save the current drawing to disk for once automatically).
- This interval value is the number of effective commands that have been executed after the last disk saving. An effective command is a command that affects the drawing database and that can be un-done by **UNDO** command.
- AXOSCALE** Real value, controls the auto-scaling function setup by the **AXOPLANE/ISOPLANE** commands. Default is 1.
- If the value of this variable is zero, then the auto-scaling function in **AXOPLANE** setup will be disabled. The length of the unit vector on each projected axis will be set to its natural foreshortening factor on that axis in the Axonometric Projection. The value of **MVSCALE** will always be reset to zero.
- However, if the value of this variable is not zero, then the auto-scaling function in **AXOPLANE** setup will be enabled. The auto-scaling function will allow the **AXOPLANE** setup to follow some drawing convention in setting up the Isometric/Dimetric planes, while keep tracking of the scaling factor for you in setting up a Trimetric skew plane.
- BEVANGLE** Double precision, angle value, default = 0°. Controls the beveling of joined wide lines (wide polyline segments). If its value is zero, it specifies a default beveling, which starts only when the corner angle is less than 90°. The beveling point is fixed and equal to that set by the angle value of 90°. If its value is positive, it specifies the corner angle below which the beveling starts. The beveling point will be determined by this angle value (it will be a point extended out from the center line end point in half of the setting angle direction). If its value is negative, its absolute value also specifies the corner angle below which the beveling starts. However, the beveling point is fixed at the original wide line's outer corner.
- BYBLKCOLOR** Integer, color value, local, default= -1 ([ByLayer]). The content of this variable specifies the actual color number used for the [ByBlock] color on a non-block-element entity. The system default is to use the [ByLayer] color. To create a block containing entities with [ByBlock] color, the user must create those entities with [ByBlock] color first. However, before those entities can be the constituents of a block, they must already exist. The earlier version of TwinCAD always use [ByLayer] color for such entities.

However, the color used for such entities in AutoCAD is never a [ByLayer] color, though it may be configured to a fixed color.

<b>CDSPFILE</b>	Filename, read only, stores the current active DSP file used by <b>TwinCAD</b> . This active DSP file is set up by the <b>.DSPFILE</b> command. A DSP file is a message display file that is designed to work with an INP file, used in <i>Playback/Recording Function</i> .
<b>CHAMDIST1</b>	Real value, specifies the first chamfer distance used in <b>CHAMFER</b> command. See <b>CHAMFER</b> command.
<b>CHAMDIST2</b>	Real value, specifies the second chamfer distance used in <b>CHAMFER</b> command. See <b>CHAMFER</b> command.
<b>CHELPPFILE</b>	Character string, contains the file name of current system help file, which is initialized by the loading of system initial file.
<b>CMDECHO</b>	Integer, On/Off status, controls whether to echo the command input when the input is from the menu, script, or TCL programming. If there is no TCL program in execution, when the command echo is turned off, then only those input from the menu or scripts be suppressed from the display. However, if the command is invoked by a TCL program, most of <b>TwinCAD</b> prompting messages will also be suppressed.
<b>CMENUFILE</b>	Character string, contains the file name of current loaded menu file. The is a read only variables, is updated by <b>MENU</b> command.
<b>CSYMLIB</b>	Filename, read only, showing the path name of current active symbol library.
<b>DISPCTRL</b>	Integer, bit-flag, local, default = 0. This variable provides some special system controls. Effective bit flag values are described below: <ul style="list-style-type: none"> <li><b>Bit 0</b> Specifies whether the logical coordinate space for the display list will be made equal to the physical window's dot space or not. If it is ON, these two coordinate spaces will be made equal, and some overhead in address mapping calculations may be reduced so that the graphic speed can be faster at the expense of less precision. If it is OFF, a homogeneous fixed logical coordinate space (device independent) will be used, which is a compatible mode for old INP file to work. This bit flag also affect the generation of display list for the <b>PRPLOT</b> command.</li> <li><b>Bit 1</b> Specifies whether to expand an <b>ARC</b> command vector into <b>LINE</b> vectors before storing it into the display list or not. If it is ON, the Graphic Runtime may expand an <b>ARC</b> command vector into successive LINE vectors before storing it. This will speed up the <b>REDRAW</b> command as well as the <b>PRPLOT</b> processing at the expense of consuming more display list memory. If it is OFF, the original <b>ARC</b> command vector will be stored directly in the display list.</li> </ul>
<b>DWGCOLOR</b>	Integer array of 16 colors, volatile, default from INI file. The content of this variable stores the 16 pen-color numbers used to override the

@**PENCOLOR** setting whenever a DWG file is loaded or created as the current drawing file.

<b>DXFIOCNV</b>	Integer, On/off status, enables or disables the conversion of DIMENSION entities from the DXF file. If it is On, the DIMENSION entities from the DXF file (after R10) will be read in and converted into <b>TwinCAD</b> 's Dimension entities. If it is Off, the DIMENSION entities will not be read, but the dimension image in the BLOCK will be read in instead. See also <b>DXFIN</b> command.
<b>EIDLEVEL</b>	Integer, specifies the listing level of <b>ID</b> command. See also <b>ID</b> command.
<b>ELLIPSEARC</b>	Integer, specifies to use True Ellipse entity for ellipse and elliptic-arc in generation. The default is 0 (OFF) to compatible with the earlier version.
<b>ENCRYPT</b>	Integer, On/off status, enables or disables the encryption of the drawing work file in subsequent disk saving. If it is turned on, then whenever the drawing is to save to disk in <b>WRK</b> file format, <b>TwinCAD</b> will ask for an encryption string, which will be used to encrypt the output data string to the disk file. See also <b>SAVE</b> command.
<b>ENTORDER</b>	Integer, bit-flag control, local, default=0. The content of this variable is used to control the access order of entities. It is a bit flag control word and is described below:  <b>Bit 0</b> Selection Set Option. If this bit is 1, a temporary order set of object will be created every time when the object <b>SELECT</b> operation is fulfilled. Objects accessed from this order set will follow the selection order from the <b>SELECT</b> operation. If this bit is 0, as a default, then no order set will be created; selected objects will be processed in the database order. This bit is effective to a selection operation only when it is set to 1 before the operation.
<b>EXPLOLIMIT</b>	Integer, local, default = 4096, specifies the maximum number of new entities that can be created by the <b>EXPLODE</b> command in exploding an associatively hatched region.
<b>EXPLOTEXT</b>	Integer, On-off control, local, default=On, controls the way a text entity will be exploded by <b>EXPLODE</b> command. If it is ON, as by default, then the text entity will be totally exploded into elementary segments, which is compatible with the <b>EXPLODE</b> operation in the earlier versions. If it is OFF, then the text will not be fully exploded into segments but rather separated into Text entities containing each single character unit, while the image created by the text feature functions and the built-in symbols will still be exploded into segments.
<b>EXTDEXACT</b>	Integer, On/off status, controls whether to extend object exactly on the boundary objects in the <b>EXTEND</b> command. The default is OFF, which means that it is the imaginary geometry of boundary objects that are taken as the boundary. If it is ON, then the extending operation will be the same as in AutoCAD, and only when the object can be extended to a point on a selected boundary object, will it be extended. See also <b>EXTEND</b> command.

<b>EXTFNTYPE</b>	Integer, stores the type identification number of extended font driver loaded by the <b>TCAM Graphic Runtime</b> . This is a read only variable.
<b>FASTLTYPE</b>	Integer, On/Off status, enables or disables the quick generation of screen line type. If it is turned OFF, the line type generation on the screen will be exactly as what is expected from the plotter output, which requires <b>TwinCAD</b> to take good care of all details of calculation, and which takes more time in the drawing regeneration. If it is turned ON, all line type generation to the screen will be done by the <b>TCAM Graphic Runtime</b> driver, which is very much faster than the exact one, however, in the expense of losing accuracy in details.  The content of this variable affects only the screen line type generation. The plotting output will not be affected by this variable setting.
<b>FILLETRAD</b>	Real value, specifies the current fillet radius used in <b>FILLET</b> command. See <b>FILLET</b> command.
<b>FILLMODE</b>	Integer, On/Off status, controls the solid color fill generation of drawing entities with solid color fill attributes. If it is ON, the solid color fill will be generated to the output devices ( screen, plotter, and other possible devices) when output. If it is OFF, then the solid color fill attribute of drawing entities will be suppressed in output, i.e., no solid color fill will be generated.
<b>HIDEANONYM</b>	Integer, On/Off status, controls the hiding of anonym objects from the selection list of names. The default is ON, which means the names of such objects will be invisible to you in the name selection.  An anonymous object is usually an object media generated by <b>TwinCAD</b> to represent something, such as a group of hatching lines in a block. The naming of such an object is given by <b>TwinCAD</b> automatically, and may not be meaningful to you (though it may, once in a while). So, hiding it from the name selection list will make you feel easier.  All the anonymous objects generated by ACAD in DXF file format will also be recognized and controlled by this variable. This will be especially benefited, since all the dimensions read from the DXF files will be in anonymous blocks.
<b>INPSTATE</b>	Integer, Bit-flag, Read only, stores the current status of the Playback/Recording Mode. See <b>TCL Command Language Reference Manual</b> in the system variable for details.
<b>INSBASE</b>	Point coordinates, specify the insertion base of the current drawing. See <b>BASE</b> command.
<b>INSDMODE</b>	Integer, controls the display of a Block instance (a reference of block). If it is 0, the block instance will be displayed in normal way (solid). If it is 1, the block instance will be drawn in dotted line, so that you can recognize it from the display at once.
<b>LIMMAX</b>	Point coordinates, specify the maximum drawing limit coordinate setting. It restricts the area of grid display.
<b>LIMMIN</b>	Point coordinates, specify the minimum drawing limit coordinate setting. It restricts the area of grid display.

<b>MAXPVANG</b>	Real value, specifies the maximum interval of arc angle allowed for the plotter output to make segmentation over a circular curve segment. See <b>PLOT</b> command.
<b>MAXPVLEN</b>	Real value, specifies the maximum length of the line segment (in real world unit of MM) used in segmentation over a circular curve segment in plotter output. See <b>PLOT</b> command.
<b>MENUTYPE</b>	Integer, controls the behavior of the pull down menu. If it is 0, which is the default, the pull down menu will not automatically drop down until the operator pick at one of the menu titles. If it is 1, the pull down menu will automatically drop down the menu whenever the cursor reaches at the menu titles.
<b>MINDONUT</b>	Real value, local variable, specifies the minimum ratio between the inner and outer diameter of AutoCAD Donut entities. When using <b>DWGIN/DXFIN</b> command to read in a Donut entity which has ratio value greater than this setting, the Donut will be converted to a corresponding entity with wide line property.
<b>MIRRTEXT</b>	Integer, on/off status, enables or disables the mirroring of Text entities. If it is On, a Text entity will be mirrored by the <b>MIRROR</b> command as if it is only a groups of lines and arcs. If it is Off, the Text entity will be mirrored in such a way that its starting position and angle of direction are adjusted so that the text writing area are mirrored. See also <b>MIRROR</b> command.
<b>MVSCALE</b>	Real value, controls the overall scale factor of the mapping view transformation. It defaults to zero, which means no scaling at all. It is effective in such commands like <b>ELLIPSE/MAP</b> and <b>MVCOPY</b> , which generates the ellipse by mapping a circle placed at a given entity plane and view from a specific direction. The transformation involves successive matrix multiplication. If the content of this variable is not zero, it will be taken as a constant scale factor to the matrix operation. The effect will be the objects being scaled in every dimension.
<b>NOEMPTYBLK</b>	Integer, On/Off control, local variable, defaults to On. specifies whether to remove empty Blocks during the read/write of drawing files.
<b>PDMODE</b>	Integer, specifies the display type of the POINT entities, ACAD-Compatible. See also <b>POINT</b> command.
<b>PDSIZE</b>	Real value, specifies the size of the POINT entities in display, ACAD-Compatible. See also <b>POINT</b> command.
<b>PICKBOX</b>	Integer, specifies the size of the object snap box in units of pixel.
<b>PICKQUERY</b>	Integer, On/Off status, controls the query function in single object picking. If it is turned ON, whenever there are multiple objects crossed by the target box, while only a single object is required, <b>TwinCAD</b> will enter the query mode for the operator to decide the one in desire. Valid only when selecting objects without any object snap directives. During the query mode, each of these possible objects will be highlighted one by one, with a query message "[?]" in the command area. The operator may press <CR> or <Pick> key to confirm the choice. Any other key will skip it for the next one until the last one which makes no choice.

If it is turned OFF, as a default, the object snapping always return the first one crossed by the target box, in the drawing database order.

- PINPFILE** Filename, Read only, stores the current active INP file used in *Playback Mode*. The Playback mode may be set up at the command line when entering **TwinCAD**, or invoked by the **REPLAY** command.
- PLAYDELAY** Integer, specifies the time delay modification count for the playback mode. This variable will be reset to zero whenever the playback mode is newly started (by **REPLAY** command). Its content will be incremented/decremented by one upon the keystroke of '-'/'+' during the playback mode. It is used as a bias to adjust the time delay for each subsequent data input read from the playback stream, units in the system tick (1/18.2 sec).
- POFSTCANG** Real angle value in units of degree, controls the offset generation of a polyline. It specifies the minimum corner angle between two adjacent segments that are allowed to have a special rounding treatment. See **OFFSET** command.
- PPLSYMBOL** Symbol name, specifies the name of the loaded symbol to represent the transparent layer of plot paper layout used in **PLOT** command for preview display purpose. If the symbol name is not specified, or the specified symbol is not loaded, the **PLOT** command will display the selected paper size in white rectangle only.
- QTEXTGENR** Integer, On/off status, controls the option of quick spline text generation. If this option is ON, all the text segments that was built with spline arc (Quadratic Bezier Curve) will be approximated each by 5 line segments, instead of the original two arcs, and that was built with Cubic Bezier Curve, will be approximated by 8 line segments, instead of the original 4 arc segments.
- The generation of such line approximation is much faster than the arc approximation, however, in the expense of less accuracy. Note that this option will not affect the result of **EXPLODE** and **PRPLOT**. Yet, the **PLOT** command is affected deliberately.
- QTEXTMODE** Integer, On/off status, controls the quick text display mode. If this mode is ON, then all the TEXT entities will be displayed as rectangular boxes at their text area.
- The use of **AUTOQTEXT** function is preferred than the use of this function. See also **AUTOQTEXT**.
- RINPFILE** Filename, Read-only, stores the current active INP file used in *Recording Mode*. The Recording Mode may be set up at the command line when entering **TwinCAD**, or invoked by the **.RECORD** command.
- SARCTYPE** Integer, controls the generation of the Spline Arcs (Dual-Arc Approximation of spline curve). The default value of this variable is 0. The content of this variable is described below:
- Bit 0-6** Bias Count if not zero, see explanation in later paragraphs.
  - Bit 7** 1 if bias backward, 0 if bias forward.

<b>Bit 8</b>	1 if find Minimum Radius Difference (No biasing), 0 if controlled otherwise.
<b>Bit 9</b>	1 if using Bisecting Angle Method, 0 if using Middle Point method (see latter paragraphs).
<b>Bit 10-13</b>	Reserved, should be zero
<b>Bit 14</b>	1 if generate complement arcs, 0 if standard.
<b>Bit 15</b>	Reserved, should be zero

If Minimum Radius Difference is enabled, then only Bit 14 is valid, and all else are ignored.

Affected commands are **SARC**, **SPLINE**, **CSPLINE**, **BSPLINE**, and the TCL function `sarc()`. Note that the **ELLIPSE** command is not affected. However, the method chosen to generate the SARcs for an ellipse approximation is changed from Middle Point to Bisecting Angle without any bias value. This change will make the approximation more closer to the ellipse when the eccentricity is very large and the spline segment number used is small.

The Dual-arc Approximation technique is a good method to approximate a given plane curves. With two given points on the curve and their tangent directions also known, the approximation will produce two arc segments tangent to each other and passing the two points at the given directions. However, there are infinite number of solutions to meet such geometry requirement, if the closest approximation to the original curve is not taken into consideration.

Different method may be used with different property of curve to determine the optimal division point (where the two arcs joint) for the best approximation. An application that uses this technique must have known the property of the curve very well, and is responsible to develop an effective method to locate the optimal points. Such an example is the **INVGEAR** command that produces the precise shape of Involute Gears.

See also ***TCL Command Language Reference Manual*** in the system variable for more informations.

<b>SAVEBACKUP</b>	Integer, On/off status, controls the feature to rename an existing <b>WRK</b> file to <b>BRK</b> (Backup-wRK-file) file before updating it.
<b>SAVEFILE</b>	Local filename, specifies the filename to be used for AUTOSAVE operation. If the filename is not specified (a null string), then the current filename will be used for AUTOSAVE.
<b>SAVETIME</b>	Integer, local variable, specifies the time interval for AUTOSAVE. The time is counted in minutes from the first UNDOable command after the last SAVE.
<b>SPLINESEG</b>	Integer, specifies the number of additional data points on a parametric curve, such as a spline curve, to be added before applying the dual-arc-segment approximation. Affected commands are <b>BSPLINE</b> , <b>CSPLINE</b> , and <b>ELLIPSE</b> commands. See also these commands.
<b>SPLINETYPE</b>	Integer, specifies the type of B-spline to generate in the <b>BSPLINE</b> command. If it is 2, the B-spline is generated in order of 3 (quadratic). If it is 3, the B-spline is (cubic) in order of 4. See also <b>BSPLINE</b> command.

- SVARORDER** Integer, specifies how the system variables will be displayed in the **SYSVAR** window.
- Bit 0:** 0, system variables displayed with system default; 1, displayed in alphabetical order.
- Bit 1:** 1, system variables displayed in the sequence defined in TCADVAR.INF file.
- When **SVARORDER = 3**, all system variables will be displayed.
- SYMESTYLE** Style name, specifies the Extended Font style name for the text used in symbols. The initial default is nothing. This is a global setting of the text style used in symbols, but is a local setting for the text display in the Attribute Tags. So, be sure to set the desired one before inserting a symbol containing attribute tags. See also **SYMBOL** command.
- SYMSTYLE** Style name, specifies the English Font style name for the text used in symbols. The initial default is STANDARD. This is a global setting of the text style used in symbols, but is a local setting for the text display in the Attribute Tags. So, be sure to set the desired one before inserting a symbol containing attribute tags. See also **SYMBOL** command.
- SYSOPTION** Integer, specifies the system operation modes of <ALT-> hot keys, defaults to zero.
- Bit 0:** ON, the <ALT-> hot keys will function based on the menu file definition. OFF, the <ALT-> hot keys will follow the Windows hot key conventions for activating the pull-down command menus.
- SYSVERSION** Integer, stores the version number of TwinCAD, read-only.
- TCAMDXFEXT** Integer, On-off control, controls whether the DXFOUT command will produce DXF file with TCAM/DXF-Extension or not. The default is OFF.
- TXTBOXGAP** Real value, specifies the default gap distance between a text and its virtual bounding box, used in **HATCH** command. If the value specified in this variable is positive, then it specifies the gap distance in drawing unit directly. However, if the value is negative, its absolute value then specifies the ratio of the gap distance with respect to the text height. If the value is zero, it means no gap at all, and the bounding box will start from the base line to the text height.
- The system default value is **-0.25**, which means the gap distance will be 1/4 of the text height.
- Note that the specification of this variable will not affect the display result of **QTEXT** mode. The **QTEXT** mode will display the text bounding box without any gap distance calculation (so as to be fast).
- TXTSUBFAC** Real value, specifies the scale factor of the text height for the text superscript and the text subscript. Default to 1. See also **TEXT** command.
- UCSCONTROL** Integer, bit flag, controls the display of an UCS-icon and other related operation, defaults to zero. Useful bit control flags are described below:
- Bit 0:** ON, disable rotation and oblique feature on UCS-Icon, valid only when the UCS-Icon is an external symbol. If

the UCS-Icon is **TwinCAD** generated vectors, it will always be rotated and oblique with the **PUCS**-Axes directions.

**Bit 1:** ON, always display the UCS-Icon at lower left corner, provided that the size of the UCS-Icon is given relative to the screen window. If an absolute size value is given for the UCS-Icon, it will always be displayed at the UCS-origin in the drawing's model space.

**Else** Reserved, should be zero.

<b>UCSORG</b>	Point coordinate, specifies the current <b>UCS</b> (User Coordinate System) X and Y origin in <b>WCS</b> (World Coordinate System). Note that setting this variable changes only the UCS origin. It does not alter the current UCS active status. To turn ON or OFF the UCS, type <Ctrl/U> command.
<b>UCSORGZ</b>	Real value, specifies the current <b>UCS</b> (User Coordinate System) Z origin in <b>WCS</b> (World Coordinate System).
<b>UCSSIZE</b>	Real value, specifies the display size of the <b>UCS</b> icon. If the <b>UCS</b> icon is a symbol reference, this variable specifies the scale factor for it if the value is positive, or a relative scale factor (percentage value) to the window width, assuming the symbol was prepared in a unit square of one drawing unit.  However, if the <b>UCS</b> icon is not specified as a symbol reference, since it will be displayed as a POINT entity, the content of this variable functions as the <b>PDSIZE</b> to a POINT entity. See also <b>PDSIZE</b> .
<b>UCSSYMBOL</b>	Symbol name, specifies the name of the loaded symbol to represent the <b>UCS</b> icon in display. If the symbol name is not specified, or the specified symbol is not loaded, the <b>UCS</b> icon will be displayed as an axes-tripod. The Projected X-axis is drawn with an arrow pointer and marked with an 'X' character at its end. The projected Y-axis is drawn with an arrow pointer only. The Projected Z-axis is drawn as a straight line. The length/size of each axis drawn in the axes-tripod shows the relative foreshortening factors between them.
<b>USE_EMODE</b>	Integer, On-off control, specifies whether to use the property from the Entity Mode Setup ( <b>EMODE</b> command) for entities created by certain editing commands. The default is OFF. The setting of this variable affects only those editing commands that will be or has been enhanced to recognize it and so as to have a useful application with the said effect.
<b>USERVARINF</b>	Filename, specifies the name of the information file for user variables. A user variable may be created by TCL function.
<b>VWNDMAX</b>	Point coordinates, read only, showing the maximum coordinates of the current view window. It is updated by the <b>ZOOM</b> and <b>PAN</b> command.
<b>VWNDMIN</b>	Point coordinates, read only, showing the minimum coordinates of the current view window. It is updated by the <b>ZOOM</b> and <b>PAN</b> command.
<b>VZCENTER</b>	Real value, specifies the Z-center coordinate of the 3D view window box. See <b>VPOINT</b> command.
<b>VZSCALE</b>	Real value, specifies an additional scale factor on the Z-axis coordinate when in 3D viewpoint. See <b>VPOINT</b> command.

<b>WORKEMAX</b>	Point coordinates, read only, containing the maximum coordinates of the current work drawing. It is updated by <b>ZOOM/Extent</b> and also new entities created.
<b>WORKEMIN</b>	Point coordinates, read only, containing the minimum coordinates of the current work drawing. It is updated by <b>ZOOM/Extent</b> and also new entities created.
<b>ZESCALE</b>	Double, local, default = 1. Used to specify an additional zoom factor to the drawing extent when the <b>ZOOM/Extent</b> command is executed. The default is 1.0, which means the view will be zoomed exactly to the drawing extent. Valid range of this value is from 0.5 to 1.0, and a value no less than 0.9 would be more desirable.

**Procedure:**

@CMD:**SYSVAR** (*return*)

(*Operate on slide window*)



# TABLET

## Tablet Device Setup Command

---

### Purpose:

The **TABLET** command is used to set up the tablet device that the system is connected to.

### Description:

The **TABLET** command lets you set up the tablet device that the system is connected to. The operations may include:

- **Device Setup** -- specifying which kind of tablet device that is in use, the hardware channel to which it is connected, and the characteristic of the hardware channel. This operation will also force **TwinCAD** to re-initialize the tablet device.
- **Tablet Configuration** -- defining the screen pointing area, the number of tablet menus to use and the corresponding tablet menu areas.

The result of these operations are tablet device parameters saved in the disk file named "**POINTER.SET**". When **TwinCAD** is started, this tablet device parameter file, "**POINTER.SET**", will be read in and used to initialize the tablet device that is supposed to be connected to the system. If this initialization is successful and the tablet device is operating well, **TwinCAD** will establish a tablet device handler that takes the tablet device as the first pointing device of the system. If you have a mouse driver resident in the memory, the mouse will be the second pointing device then.

When you enter the **TABLET** command, **TwinCAD** will check if a valid tablet device handler had been established or not. If the handler is not yet established, you will be asked to do Device Setup first before initializing the tablet device. On the other hand, if it is already well established, **TwinCAD** will then prompt:

```
Tablet -- Device-setup/<Configuration>:
```

for you to choose between these two operations. If you press the space bar or reply it with a null return as a default, **TwinCAD** will let you do the Tablet Configuration. You may request to do the Device Setup by entering the sub-command option "**D**".

### Tablet Configuration

As the tablet device usually operates on absolute mode, i.e., it reports the absolute coordinate of the pointer on the tablet surface, the system can easily identify the position of the pointer on the tablet. It is therefore possible to define some areas on the tablet to be meaningful to **TwinCAD**. When you digitize (press the button on the digitizer, the pointer) a position in these areas, **TwinCAD** will know exactly what you intend to do. This is where the concept of the tablet menu stems from.

Therefore, in addition to the basic area on the tablet which is required for screen pointing purpose, you may configure some more areas on the tablet to be used as tablet menus.

**TwinCAD** supports at most 4 rectangular tablet menu areas. Each of these areas can be further sub-divided into smaller rectangles by the number of columns and rows. The maximum number of this sub-division for a tablet menu is 512. The size and the location for these tablet menu areas and the screen pointing area are arbitrary, as long as they do not overlap.

At the start, **TwinCAD** will ask for the number of tablet menu areas you desire, with the prompt:

Enter number of tablet menu desired (0-4)<n>:

If you have specified the number of tablet menu to use, and it is not 0, then **TwinCAD** will ask you to indicate where these menu areas are, one after another, with the prompt in sequence for each of them:

Digitize upper left corner of menu area *n*:

Digitize lower left corner of menu area *n*:

Digitize lower right corner of menu area *n*:

Enter the number of columns for this area <colno>:

Enter the number of rows for this area <rowno>:

where the letter *n* represents the menu area number from 1 to 4. Note that the product of the column number and row number must not exceed the limit of 512.

After setting up the tablet menu areas, **TwinCAD** will ask you to specify the screen pointing area, with the prompts in sequence:

Digitize lower left corner of screen pointing area:

Digitize upper right corner of screen pointing area:

To verify that **TwinCAD** can read the tablet device, when you are asked to digitize a particular position, you can read the coordinate values from the tablet being displayed at the status line.

Note that some tablet device may be configured to operate in Delta Mode, i.e., it will report the relative movement of the pointer like mouse, and **TwinCAD** will respond

\*\* Device operates in delta mode (mouse mode).

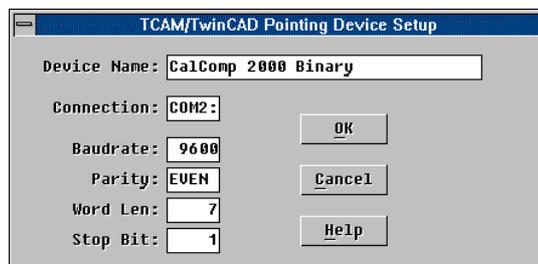
and refuse to go on the Tablet Configuration operation.

## Device Setup

**TwinCAD** will pop up a Pointing Device Setup Window for you to do the setup operation by

- Selecting a proper **Digitizing Device Setup File (DDS)** for the tablet device,
- Specifying which communication port to be used for the tablet device,
- Specifying the communication protocol such as the baud rate, parity, word length and stop bit number.

These are described below:



### Device Name and the Digitizing Device Setup File (DDS)

The *Digitizing Device Setup file* (or **DDS** in short) is a file containing parameters of a particular type of tablet device. **TwinCAD** provides several DDS files, well defined for some renowned

tablet devices. Each DDS file is responsible for a type of tablet device, of which the name is also included in the file. The field of device name thus shows the name read from the selected DDS file.

To set up the proper device name for the tablet you are using, simply pick up the corresponding screen item. **TwinCAD** will search the disk for all the DDS files and pop up a device name selection window for you to select the proper one by the name provided in these DDS files.

Should there be no DDS file found from the current logged disk or from the path specified by the system environment variable **TCADPATH**, **TwinCAD** will then open the file window for you to directly select the DDS file.

Once a DDS file is chosen and read in, the hardware communication protocol will be reset by the default setting from the DDS file.

You can use the utility **DDS.EXE** to build the DDS file for your tablet device, if you could not find a proper one from those shipped with the package. See the Utility part of this document for details.

## Communication Port and Protocol

**TwinCAD** supports 4 communication ports from **COM1:** to **COM4:**. Simply pick up the field showing the port name, and then you can select the desired port name from the pop-up selection window.

To change the baud rate, word length, parity and stop bit number, also pick up the corresponding screen field, and then you can change it as you wish.

### Reinitialize the Tablet Device

After proper setup of these device parameters, you may pick the [ OK ] button in the window. **TwinCAD** will re-initialize the tablet device using the new parameters. If the tablet device does not work, the message

\*\* Digitizer not respond!

will be reported. If you pick the [Cancel] button, the original setup parameters will be read back and the tablet device will be reinitialized again.

So, if you have just forgotten to turn the tablet device power ON before entering **TwinCAD**, you may turn it ON later and issue the **TABLET/D** command to enter the Device Setup window, and then cancel the operation, such that the tablet device will be re-initialized again.

### The Parameter File: "POINTER.SET"

**TwinCAD** stores the setup parameters of the tablet device in the disk file: "**POINTER.SET**". During **TwinCAD** initialization, this file will be read in and used to initialize and configure the tablet device. After properly setting up the tablet device, you may backup this parameter file to a safe place, so that you can restore it back if the setup should have been changed accidentally.

The structure of this parameter file is the same as the DDS file. In fact, what **TwinCAD** does, when you specify the tablet device by selecting the DDS file, is to make a copy of it to the file "**POINTER.SET**". So, during the installation of **TwinCAD**, the default content of this "**POINTER.SET**" is a copy of a selected DDS file.

To remove the setup of a tablet device completely, just delete this parameter file.

**Procedure:**

- To configure the tablet for 2-tablet menu layout  
@CMD **TABLET** (*return*)  
Tablet -- Device-setup/<Configuration>: (*space bar*)  
Enter number of tablet menu desired (0-4)<n>: **2**  
Digitize upper left corner of menu area 1: (*do so*)  
Digitize lower left corner of menu area 1: (*do so*)  
Digitize lower right corner of menu area 1: (*do so*)  
Enter the number of columns for this area <colno>: **n1**  
Enter the number of rows for this area <rowno>: **n2**  
Digitize upper left corner of menu area 2: (*do so*)  
Digitize lower left corner of menu area 2: (*do so*)  
Digitize lower right corner of menu area 2: (*do so*)  
Enter the number of columns for this area <colno>: **n1**  
Enter the number of rows for this area <rowno>: **n2**  
Digitize lower left corner of screen pointing area: (*do so*)  
Digitize upper right corner of screen pointing area: (*do so*)

**Example:**

# TAGEDIT

## Edit Attribute Tag Content Command

---

### Purpose:

The **TAGEDIT** command is used to edit the content of attribute tags of selected objects or of block definitions.

### Description:

The **TAGEDIT** command lets you edit the content of the attribute tags of selected objects or of block definitions. The attribute tags to the whole drawing (in work drawing file level) can also be edited by this command.

The attribute tags of a block definition are given with initial values when the block is being defined (see also **BLOCK** command). When a block instance is created, local attribute tags are also created with these initial values, and are edited immediately for the purpose of local entries (See also **INSERT** command). After that, the only way to change the current values of these attribute tags is to use this **TAGEDIT** command.

The attribute tags in work drawing file level are tags that attach to no entities but the whole drawing. Such attribute tags will become private attribute tags when the whole drawing is loaded as a block. They are useful and essential in auto-management of work drawing, since utility programs may directly access predefined attribute tag and determine the management status of the drawing. Currently, there is no direct command support for users to create the attribute tags at drawing file level. Nevertheless, there are TCL interface functions which will support this usage. Suppose you want to create such tags without going through the TCL programming, you may directly issue the TCL function **add\_tag()** from the command line, with the arguments supplied explicitly. See ***TCL Command Language Reference Manual*** for details.

When you enter the **TAGEDIT** command, **TwinCAD** will prompt

```
Tagedit -- Block/Drawing/<Select block instance or symbol>:
```

asking you to pick up the objects of which the attribute tags are to be edited. When you pick up an object that is not tagged with attributes, **TwinCAD** will complain with the message

```
** Object does not contain any tag.
```

and continue the prompt again. Once a tagged object is picked, the system will pop up the Attribute Tag Editing Window for you to edit the attribute tags.

You may enter the sub-command option "**B**" to edit the attribute tags of a block definition. **TwinCAD** will pop up a slide window for block name selection. You simply select the block by its name from the window and then the Attribute Tag Editing Window will also be popped up for you to edit the initial values of these attribute tags that follow the block definitions.

If you enter the sub-command option "**D**" to the prompt, **TwinCAD** will let you edit the attribute tags at the work drawing file level. The Attribute Tag Editing Window will also be popped up. Note that if there is no attribute tag at the work drawing file level, the pop-up window will be empty.

Note that you can't edit the constant attribute tags of a selected object or a work drawing file. While you may edit the constant attribute tags of a block definition. The constant attribute tags

of a block definition are global to all block instances that reference to the same block definition, and therefore can only be edited directly from the block definitions.

## The Attribute Tag Editing Window

The Attribute Tag Editing Window provides operations to enter, view, and modify the contents of attribute tags in a tabulated manner, and is popped up

- when a block is being defined and an attribute tag group is chosen, for initial value entries of the attribute tags so defined, or
- when a block instance or a symbol is being created and the original definition contains variable attribute tags, for value assignment or modification of the local attribute tags so created, or
- when the **TAGEDIT** command is issued and a block definition, block instance or symbol that contains tags is selected, for modification of the current content of these attribute tags.

Each attribute tag in the window is tabulated as a line containing

- **DV status** -- Display and Variable status. When a mark of '\*' is placed at this field, it indicates that the content of the attribute tag is viewable and will be displayed. A mark of '?' will indicate that the attribute tag is a variable one.
- **Tag Item Title** -- A prompting title for the attribute tag. The prompting title of an attribute tag is defined by the **TAGVAR** command, and is used to tell the operator what the attribute tag is. If there is no such prompting title defined in the **TAGVAR** table, or if it is a null string, then the attribute tag name will be used instead. Note that the prompting title can be in any language that the system supports.
- **Current Content** -- Current content of the attribute tag. This field displays the current content of the attribute tag. If the attribute tag is a constant, the color of the character display will be dimmed a little bit.

Each line of attribute tag from the window is a selectable item. To modify the content of an attribute tag, just pick up its item of line by the cursor. A value entry field will be opened at the field of its Current Content and then you can modify it by Editing its current content.

If the window is popped up for initial entries of the attribute tags (especially when a block instance is being created), each attribute tag, tabulated as a line of item, will appear in sequence for value entry automatically one after another. This helps you quickly entering desired values without doing pick-and-modify operation. You may end this automatic process by returning an empty string to the attribute tag.

There are three screen buttons in the window, as shown in the figure below:

- **[All-Item]** -- request to view all the attribute tags in the window. All attribute tags will be displayed in their original sequence of creation. This is the default for block definition.
- **[Var.Only]** -- request to view only variable attribute tags in the window. All constant attribute tags are removed from the display. This is the default for block instance, since you can't edit the constant attribute tags.
- **[OK]** -- request to exit the Editing window. You may also pick up the [-] screen button or press <ESC> key to exit the window.

### Procedure:

- To edit the attribute tag of a block instance:

@CMD **TAGEDIT** (*return*)

Tagedit -- Block/Drawing/<Select block instance or symbol>: (*Select one*)  
(*Operate on window*)

- To edit the attribute tag of a block definition:

@CMD **TAGEDIT** (*return*)

Tagedit -- Block/Drawing/<Select block instance or symbol>: **B** (*return*)  
(*Select block*)  
(*Operate on window*)

**Example:**

# TAGIN

## Load Attribute Tag Definition File Command

---

### Purpose:

The **TAGIN** command is used to define attribute tags and attribute tag groups by loading in a particular attribute tag definition file.

### Description:

The **TAGIN** command lets you load in a particular attribute tag definition file from disk and thereby define the attribute tags in the **TAGVAR** table and the attribute tag groups.

You may use the **TAGVAR** command to define the attribute tags that will be used in the drawing. And then, use the **TAGS** command to create the tag groups that help specify these attribute tags in creating a block definition. Though it is convenient to apply attribute tags to a block definition in this way, it may still be tedious and time-consuming to set up the tables each time a new drawing is created. See **TAGVAR** and **TAGS** commands for related information.

To make your job easier, you may prepare a disk file which contains the definitions of attribute tags and tag groups in advance, and use the **TAGIN** command to load in the pre-defined tag definition file each time a new drawing is created. It is not necessary to prepare such tag definition file with a text editor. You may use the **TAGVAR** and **TAGS** commands to set up these attribute tags at the first time, and then use the **TAGOUT** command to save it out as a tag definition file for later use.

### The Attribute Tag Definition File (TAG file)

The *Attribute Tag Definition file*, or **TAG** file in short, is a text file containing the required information to define the attribute tags as well as tag groups in **TwinCAD**. The TAG file may contain lines of comments, tag definitions, and tag group definitions.

Any line with a ';' (semi-colon) as the first non-blank character is a line of comment and will be ignored by the **TAGIN** command. Actually, any character after the ';' (semi-colon) in a line will be taken as a comment.

### Tag Definitions

A line of tag definition is used to define an attribute tag, and is in the form:

```
TAG: Tag-name, {D,V}, "prompting title"
```

where,

- **TAG:** -- Tag definition indicator, indicating a line of tag definition, must be the starting string of the line. Leading blanks are removed and character cases are insensitive.
- **Tag-name** -- Name of the attribute tag, maximum 10 characters, will be converted to upper case. Valid characters for this tag name are alphanumeric, '\_' (underscore) and '\$' (dollar sign).
- **D** -- Optional display status, indicating the attribute tag will be viewable.
- **V** -- Optional variable status, indicating the attribute tag is a variable tag.

- *"string"* -- Prompting title in quoted string, used in the Attribute Tags Editing Window, can be of any language codes.

The rest characters of the line are ignored. Note that the proper delimiters between these items are commas. Should the parsing encounter an error, **TwinCAD** will issue the message:

Error at line nn: Illegal characters.

and then ignore the line.

## Tag Group Definition

Tag group definition is used to define a tag group, and may occupy several lines. It is expressed in the form:

```
GROUP: groupname, "Usage note" { tag-name1, tag-name2, ... }
```

or in the form

```
GROUP: group-name, "Usage note"
{
    tag-name1, tag-name2, ...
    ...
    tag-namen,...
}
```

where,

- **GROUP:** -- Tag group definition indicator, indicating the start of a tag group definition, must be the starting string of the line. Leading blanks are removed and character cases are insensitive.
- *group-name* -- Name of the tag group, maximum 10 characters, can be any codes of language except the comma and space.
- "*Usage note*" -- Usage note in quoted string, can be any codes of language.
- { -- Left brace, indicating the start of the tag name in the group.
- *tag-namen* -- Name of the attribute tag that is quoted in the group. The maximum number of tag-name in a group is not limited. The order of appearance of each tag-name will be the order in the group.
- } -- Right brace, indicating the end of the tag name in the group.

Note that the proper delimiters between these items are commas.

The order of appearance of each tag definition and tag group definition is irrelevant. However, system will send error message for forward reference of tag-names in tag group definition:

Error at line *nn*: Ignore forward reference of tag name: *xxxxxx*

## Action for Possible Name Conflict

It is possible to load a TAG file containing existing attribute tag definitions and tag group definitions. In such a case, there are three possible actions to solve this name-conflict problem:

- **Ignore:** the newly loaded one is ignored.
- **Override:** the newly loaded one replaces the existing one.
- **Merge:** the newly loaded one merges with the existing one (tag group definition only)

When you issue the **TAGIN** command, **TwinCAD** will prompt

Select action when names conflict -- Ignore/Merge/<Override>:

asking you to select the desired action. Note that if you select to merge in the TAG file, the name conflict in attribute tag definition will be solved in Override Mode, which is also the default.

The sub-command option "**I**" is used to specify the action to ignore the newly loaded attribute tag whose name is in conflict with an existing one. The sub-command option "**M**" is used to specify the action to merge in the definition of tag group, if there is such name-conflict encountered.

The file window will be popped up for the selection of TAG file from the disk.

### Procedure:

@CMD: **TAGIN** (*return*)

Select action when names conflict -- Ignore/Merge/<Override>: (*as you wish*)

(*Select file on file window*)

### Example:

An example content of tag file is given below:

```

;      =====
;
;      Title:  Example of a Tag File
;
TAG:PARTNAME ,D,"Enter Partname"
TAG:MATERIAL ,V,"Material Used"
TAG:ORIGIN   ,V,"Made From"
TAG:PRICE    ,V,"Unit Price"
TAG:MODEL    ,V,"Product Model"
GROUP: Product,"General Product"
{
    PARTNAME,
    MODEL,
    ORIGIN,
    MATERIAL,
    PRICE
}

```

# TAGOUT

## Save Attribute Tag Definition to File Command

---

### Purpose:

The **TAGOUT** command is used to save the current definition of attribute tags and attribute tag groups into a disk file for future loading purpose.

### Description:

The **TAGOUT** command lets you create a TAG file by saving all the current definition of attribute tags and tag groups into it. The TAG file may be loaded later in other drawing by the **TAGIN** command. See also **TAGIN** command for further details about TAG file and its usage.

When you issue the **TAGOUT** command, **TwinCAD** will pop up a file window for you to specify the output TAG file name. Note that the file extension for a TAG file is always **".TAG"**.

### Procedure:

@CMD: **TAGOUT** *(return)*  
*(Specify output file on file window)*

### Example:

# TAGS

## Define Attribute Tag Group Command

---

### Purpose:

The **TAGS** command is used to define attribute tag groups.

### Description:

The **TAGS** command lets you define attribute tag groups. An attribute tag group is a list of attribute tags that will be used in sequence to apply to a block definition. When you are creating a block definition and want to attach attribute tags to it, with the aid of this attribute tag group, you won't need to create or reference any attribute tag individually -- you simply choose the proper tag group and the job is done.

A complete attribute tag group includes three parts:

- **Group Name** -- Name of the tag group for formal reference, maximum 10 characters of any language codes.
- **Usage Note** -- Usage description or comment, telling the operator what kind of tag group it is.
- **Number of Tags** -- Number of attribute tag names included in the group.

When you enter the **TAGS** command, **TwinCAD** will pop up an Attribute Tag Group Control Window. Each line in the window shows the defined tag group name, number of tag names in the group, and the usage note of the group. You may create new tag group, add/modify the list of tag names in the group, and modify the usage note as described in the following paragraphs.

To exit the window, press <ESC> key or pick up the [-] button.

### Create New Tag Groups

To create a new tag group, pick up the **<NewTagGrp>** item from the group name field. **TwinCAD** will open the field and let you enter the name of the new tag group. After that, an empty tag group is created with a null string in the usage note. You may further add attribute tag names to the group.

### Add/Modify Tag Names in Tag Groups

To add or modify the tag names included in a particular tag groups, pick up the name of the tag group directly. **TwinCAD** will open an Attribute Tag Group Setup Window for you to add or to remove specific attribute tag names to or from the group.

### Modify the Usage Note of Tag Groups

To modify the usage note of a particular tag group, pick up the usage note field of the tag group, and **TwinCAD** will let you edit the field.

### The Attribute Tag Group Setup Window

The Attribute Tag Group Setup Window is popped up for you to edit the list of the attribute tag names that a selected tag group includes. **TwinCAD** employs a tag pointer to facilitate the

editing operation. The attribute tag name pointed by the tag pointer will be highlighted, for easy tracking of the place where the tag pointer points to.

Initially, the tag pointer points to the first tag name in the list. To change the tag pointer to point to a particular tag name, directly pick up the tag name with the cursor.

There are two screen buttons in the window that provide the operations for adding new tag names and removing existing tag names, respectively:

- **[ AddNew ]** -- Request to add new attribute tag name into the list before the one pointed by current tag pointer. **TwinCAD** will pop up another Attribute Tag Selection Window for you to select the one to add into the list. The current tag pointer will not be changed.
- **[ Remove ]** -- Request to remove the attribute tag pointed by current tag pointer. The current tag pointer will move to the next entry after the deletion. If the tag pointer is already at the last tag entry before the deletion, then it will again point to the last tag name as the deletion is done.

To append new tag name at the end of the list, you have to pick up the item **<AddNewTag>** in the tag name field. The operation is the same as that after you pick up the [AddNew] button, except that the new tag name is added at the end of the list. This will not affect the current tag pointer.

To exit the Attribute Tag Group Setup Window, press **<ESC>** or pick up the [-] button or the rightmost button of mouse.

### Procedure:

@CMD: **TAGS** (*return*)  
(*Operate on window*)

### Example:

# TAGVAR

## Define Attribute Tag Items Command

---

### Purpose:

The **TAGVAR** command is used to define attribute tag items.

### Description:

The **TAGVAR** command lets you define attribute tag items. An attribute tag item is an entry in the tag variable table, which defines an applicable attribute tag including its formal reference name, functional status and the prompting title string when it is referenced.

The tag variable table is used in the **TAGS** command that creates attribute tag groups containing list of attribute tag items taken from the table. It is the attribute tag group that the operator chooses to generate attribute tags to a block definition. **TwinCAD** creates the attribute tag entities to be included in the block definition in accordance with the list of the attribute tag name from the specified tag group. See also **TAGS**, **BLOCK** and **INSERT** commands for related information.

When an attribute tag entity is created, it will assume the status from its corresponding entry in the tag variable table. When the Attribute Tag Editing Window is popped up (see **TAGEDIT** command), the prompting title of the attribute tag is also taken from its corresponding entry in the tag variable table. Changing the status of the attribute tag item from the tag variable table, however, will not change the status of these already created attribute tag entities.

When you issue the **TAGVAR** command, **TwinCAD** will pop up the Attribute Tag Variable Control Window for you to add new attribute tag items or to modify existing attribute tag items.

### Add New Attribute Tag Items

To add new attribute tag items into the table, pick up the item **<NewTagID.>** from the window. **TwinCAD** will open an entry field at the same position for you to enter the name of the new attribute tag. Valid characters for tag name are alphanumeric, '\_' (underscore) and '\$' (dollar sign). The name string will be converted into upper case. If the new tag name is not valid, or it already exists, **TwinCAD** will reject the creation of the new tag item.

Once the new tag item is created and added in the tag table, you may further modify its data status and prompting title string.

### Modify an Attribute Tag Item

You can modify the prompting title string of a selected attribute tag item by directly picking up the Prompt Title field of it. The system will open the field and let you edit its content.

You can alter the functional status of a selected attribute tag item by directly picking up the status field of it. The functional status of an attribute tag determines the usage and behavior of the attribute tag.

- **Viewable vs. Non-Viewable**

The entry under the field "DISP", indicates the display status of the attribute tag. When DISP is "YES" (ON), the content of the attribute tag entity will be viewable and displayed as a TEXT entity in the drawing. When DISP is "NO" (OFF), however, the content of the attribute tag will become hidden information in the drawing. During the creation of "viewable" attribute tag entities,

**TwinCAD** will prompt for information input as you are generating Text entities, such as starting position, text style, text height and so forth.

- **Variable vs. Constant**

The entry under the field "Type", indicates the data status of the attribute tag. When this entry shows "Var.", the content of the attribute tag entity will be a variable, and thus will be locally created and maintained for each block instance. You may assign different values to different copies of the block instances that reference to the same block definition. When the entry shows "Const.", the content of the attribute tag entity will be a constant value for the block definition. So for each block instance, you will not need to assign local value for the attribute tag. The constant value must be given when the block is being defined.

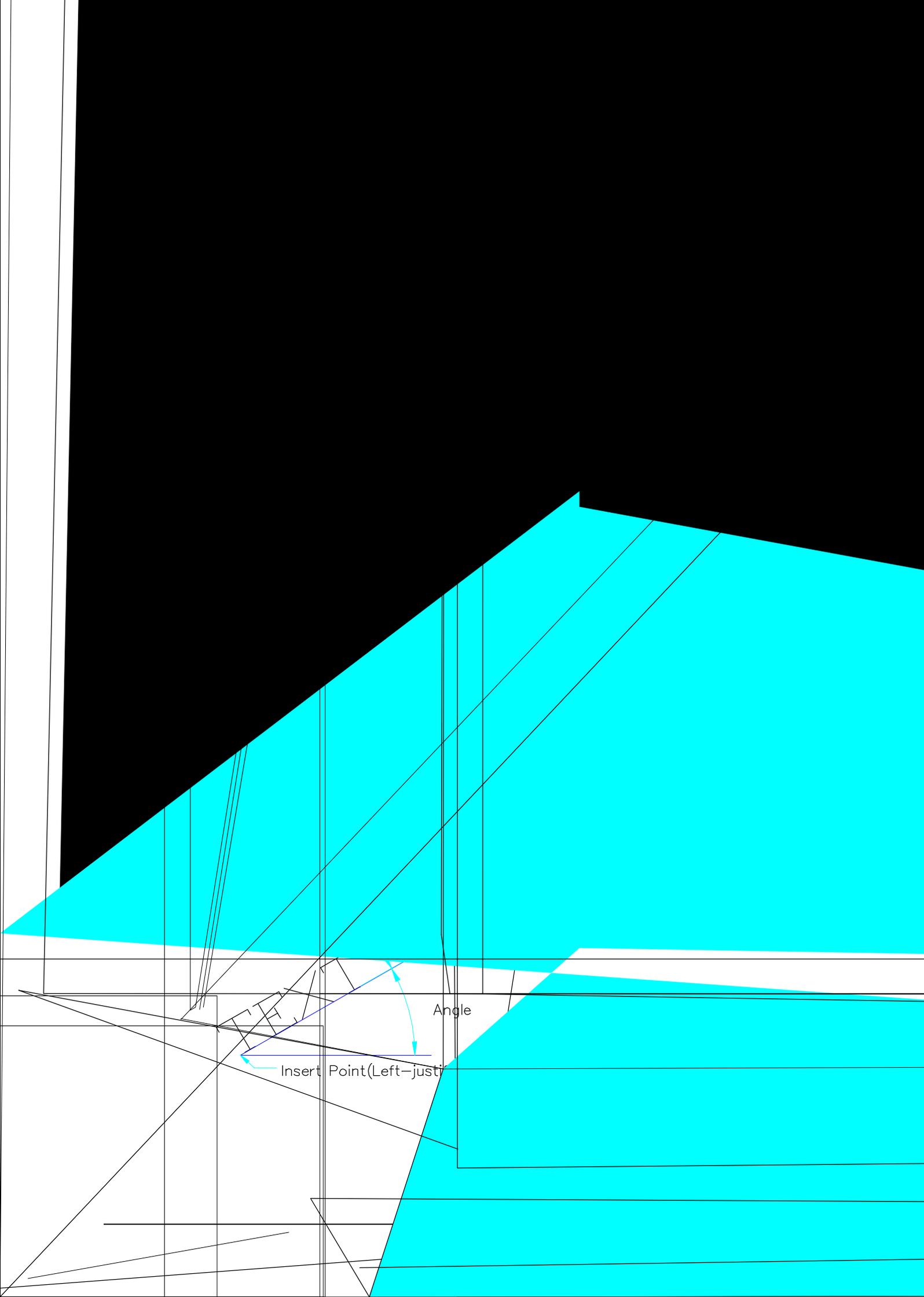
Note that you can't modify the name of an existing attribute tag. If you should do so, use the **RENAME** command. See **RENAME** command for details.

You may press <ESC> key, or the rightmost mouse button to exit the window.

**Procedure:**

@CMD: **TAGVAR** (*return*)  
(*Operate on window*)

**Example:**



Angle

Insert Point (Left-justi

To the last prompt for angle input, you may designate a center point instead, to specify that the generation of text will be in circular fashion around the center from the insert point. In such a case, the system will continue to prompt

Direction CCW/CW <xx>:



for you to specify the writing direction of the text with respect to the given center.

Finally, you will be asked to enter the text. **TwinCAD** will open a one-line text window for you to enter the character string and edit the string as well. There are many possible ways to enter the text string to create, see later section: **Entering and Editing the Text String**.

If the text string returned is a null string, the text will not be created and the command will exit. In other cases, it will be created and **TwinCAD** will continue to ask you to enter more text strings to create. The subsequent texts created will have the same text style parameters (including the angle direction or center position) as the first one, each with the insert point advanced to the next text row. The distance between text row is also controlled by the text style parameter **XTROWDIST** (see **STYLE** command).

Note if the insert point of the new text to create is determined by default (you reply with a null return), it will be assumed to continue from the Last Text creation, and you will be directly asked to enter the text string. All other text parameters will be taken from the Last Text.

The text newly created will become the Last Text for the subsequent text creation.

## Text Writing Styles

**TwinCAD** provides two types of text writing style:

**Linear writing** The text is written along a line at a given angle. You must specify the angle value in degrees. Each of the text characters may be rotated 90° counter-clockwise when writing.

**Arc-aligned writing** The text is written along a circular path around a given center, clockwise or counter-clockwise. You must designate the center point of the circle. The radius is calculated as the distance from the text insert point to the designated center. Each of the text characters may also be rotated 90° counter-clockwise when writing. The text so written will be stretched as if you have bent them from a line to a circle.

The text in arc-aligned writing is not ACAD-compatible. If you want to transform it to DXF file format, you will get a text of linear writing at an angle tangent to the circle at the text insert point. However, you may explode it to become a part of drawing.

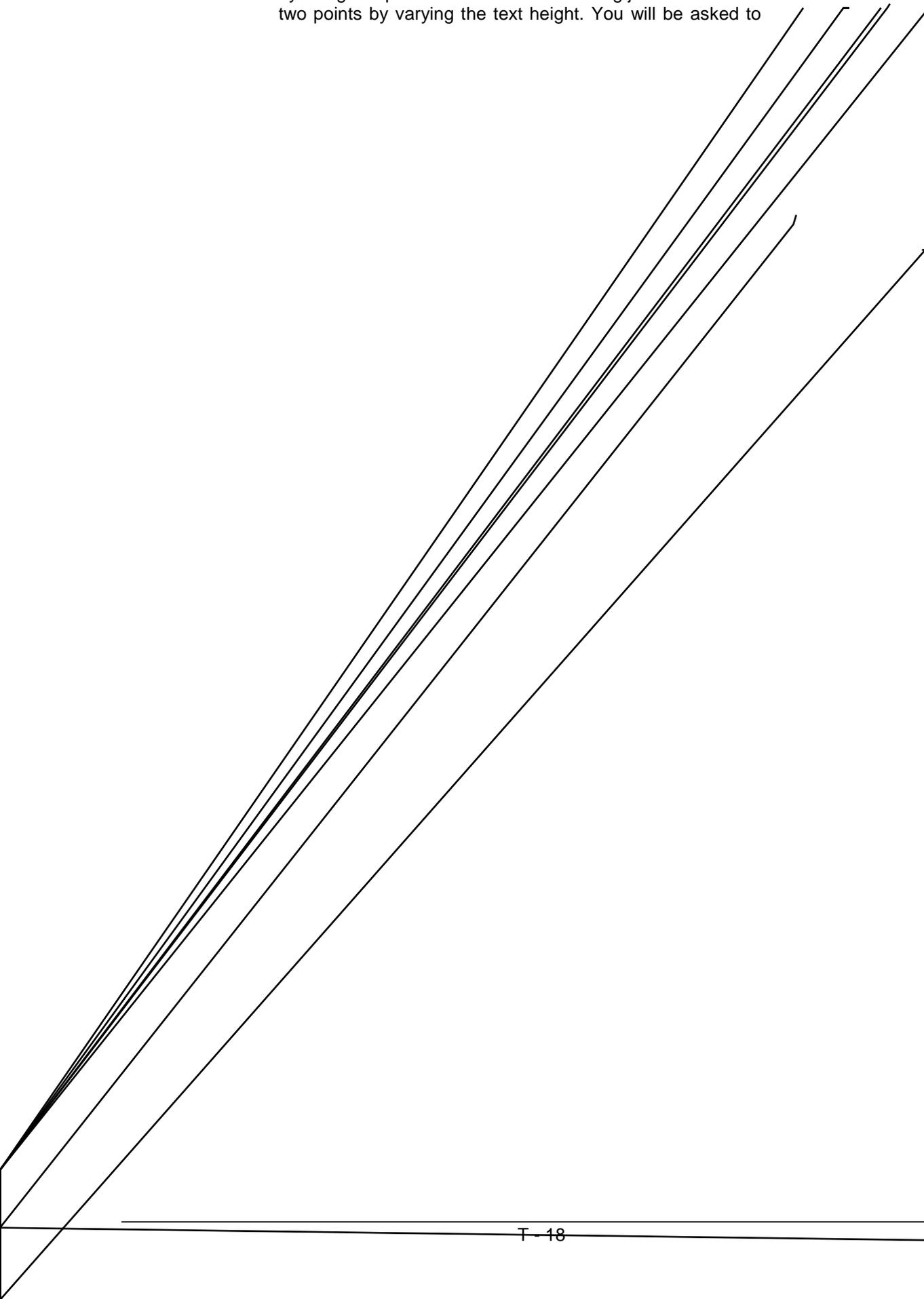
## Justification of Text

You may specify to align the text using a specific alignment mode by entering the sub-command option "J", which stands for "Justify", before designating the insert point of the text at the first prompt of the **TEXT** command. **TwinCAD** will prompt

Align/Fit/Even-space/Center/Middle/Right/<Left>:

for you to specify the alignment mode by the following sub-command options:

- A** **Align**, specifying to align the text in the direction indicated by two given points and fit in the text string just between the two points by varying the text height. You will be asked to

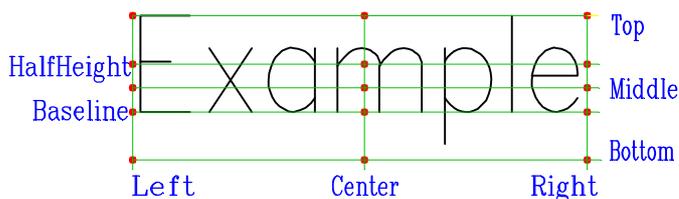


You must enter null string to the text string input to end the command.

The rest of sub-command options for text Justification will set the required alignment mode and return you to the first prompt where you may continue the text creation operation:

- space** Reset to Left Justification. You don't need to enter the "Left" but press the space bar to reset the alignment mode of text to Left Justification.
- C** **Center**, specifying to center the text at the base line. The insert point of the text will be on the base line and at the center of the text.
- M** **Middle**, specifying to center the text both horizontally and vertically at the middle of the text. The insert point of the text will be on the middle line and at the center of the text.
- R** **Right**, specifying to right-justify the text at the base line. The insert point of the text will be on the base line and to the right of the text.
- TL** **Top-Left**, specifying to left-justify the text at the top line. The insert point of the text will be on the top line and to the left of the text.
- TC** **Top-Center**, specifying to center the text at the top line. The insert point of the text will be on the top line and at the center of the text.
- TR** **Top-Right**, specifying to right-justify the text at the top line. The insert point of the text will be on the top line and to the right of the text.

### Text Alignment Points



- ML** **Middle-Left**, specifying to left-justify the text at the middle line. The insert point of the text will be on the middle line and to the left of the text.
- MC** **Middle-Center**, specifying to center the text at the middle line. The insert point of the text will be on the middle line and at the center of the text.
- MR** **Middle-Right**, specifying to right-justify the text at the middle line. The insert point of the text will be on the middle line and to the right of the text.
- BL** **Bottom-Left**, specifying to left-justify the text at the bottom line. The insert point of the text will be on the bottom line and to the left of the text.
- BC** **Bottom-Center**, specifying to center the text at the bottom line. The insert point of the text will be on the bottom line and at the center of the text.
- BR** **Bottom-Right**, specifying to right-justify the text at the bottom line. The insert point of the text will be on the bottom line and to the right of the text.
- HL** **Half-height-Left**, specifying to left-justify the text at the half-height line. The insert point of the text will be on the half-height line and to the left of the text.
- HC** **Half-height-Center**, specifying to center the text at the half-height line. The insert point of the text will be on the half-height line and at the center of the text.
- HR** **Half-height-Right**, specifying to right-justify the text at the half-height line. The insert point of the text will be on the half-height line and to the right of the text.

Note that these text justifications are also valid for circular text generation.

### Align/Fit/Even-fit of Circular Text

The sub-command options to fit in a text string between two points (Align, Fit or Even-fit) will accept a null return at the prompt for the first point to indicate that the fitting will be aligned with a given entity. After the null return, **TwinCAD** will prompt

Select line, arc or circle:

asking you to pick up the entity to be used for the base alignment. You may select a line, an arc or a circle. The text is then generated using the selected entity as its base line, with text fitting in Align, Fit or Even-fit respectively. Note that the start point of the text will be the start point of the selected line or arc, or the pick point of the selected circle.

This is the only way to generate circular texts with fitting requirement.

## Entering and Editing the Text String

When you are asked to enter the text string (or to modify the text string by **CHANGE** command), **TwinCAD** will open a one-line text window at the command area after the prompt:

Text:

You may enter the character string and edit the string as well in this one-line text window by using the editing function described in the General Informations Part of this document.

You must terminate the text string input by the return key.

If you want to enter the text in other languages than the English, then depending on the version, initial setup and optional packages purchased, you may press the **<Ctrl/Fn>** key and choose the one you want. For example, the default **Chinese Phonetic Input Window** is invoked by depressing the **<Ctrl/F1>** key. See the **Installation Guide** for related informations.

Note that the initial setup may specify to enter a specific text input handler automatically other than using the standard one described here. However, you may switch it back by **<Ctrl/F0>**. Usually, these input handlers should also support these editing functions described above.

The switching between different handlers by **<Ctrl/Fn>** is controlled by the current active input handler. Some input handlers may not recognize this request, but it must provide a way for you to exit the handler without terminating the text string input. In such a case, the standard input handler will be switched in to continue the text input operation.

## The Text Input Handlers

A Text Input Handler is a small application program developed to handle text input of different kind of languages or to implement different methods of input. Some of the text input handlers are shipped with the package you have purchased, while some are optional. Please refer to the Appendix for a list of Text Input Handlers available when this document is released.

## String Expression

The text string obtained from the text entry operation as described in previous section will be taken as an expression that evaluates to a string, if and only if it begins with the characters **"@!"**. If so, the string from the third character will be evaluated as an expression that returns a string.

If the evaluated expression contains an error or it does not return a string result, **TwinCAD** will repeat the text entry operation for you to edit the expression.

An example of such expression in text string is given below:

```
@!"Drawing Created at "+date$()
```

which will be evaluated to

"Drawing Created at Wed Jul 07, 1993"

assuming it is the date when the expression was entered. See the **TCL** Command Language Reference Manual for more details about the expression.

## Text Insertion in Continuous Mode

The text insertion will be taken as in **Continuous Mode** under the following circumstances:

- A null reply is returned by the operator for the text start point, while there is an active Last Text. The text insertion is then considered to be continued after the Last Text.
- You have indicated the start point by picking at an existing Text entity via the **INSert** snap directive. The picked Text then becomes the Last Text after which the text insertion is considered to be continued. All the text control informations, except the text styles, will be read and set to the current style table.
- A new text is just created and the text insertion is continued by its nature of operation. The newly created text is the Last Text for the next text insertion.

If the text insertion is in **Continuous Mode**, only the text string will be asked for, and all other text control informations are taken from the current style table. The start point for the new text insertion will be determined automatically by the start of the next Text Line after the Last Text.

If the Last Text is Aligned, Fitted or Evenly-Fitted, so will be the newly created text. However, if the Last Text is Aligned, then the next text line for the new text string will be determined by the text height of the new text, which is determined by the length of the text string in Align mode.

If the Last Text is in Circular Writing, the new text will also be in Circular Writing about the same center. However, depending on the direction of the text, the base circle of the writing may shrink toward the center or expand from the center. If the base circle should shrink toward its center, it will expand out again and the text writing will be generated as if the text had been mirrored with respect to the center. It will be so adjusted that the text writing should never cover its rotation center and the inner loop of texts will look balanced under the Continuous Mode.

## Feature Functions and Special Codes

You may enter any characters except the null code (00) to the text string for the text entity in creation. But, the character codes in the range from 01H to 1FH are reserved for special purpose. Besides these control codes, some character codes are also served as special function codes in the text string. These will be described in the following sub-sections.

### Special Feature Functions

Some codes are used to control the text generation with additional features, such as generating the underline over a part of the text string, superscript and subscript of text string. These are feature function codes and are described below:

'~' **Tilde**, used to enable or disable the extension font. When the use of extension font is disabled, all the font of the characters will be taken from the standard font style until the use of extension font is enabled again.

$\phi 10.50^{+0.10}_{-0.15} \text{MM}$



TEXT GENERATED BY  
"~Φ10.50{+0.10^-0.15}MM"

Initially, the use of extension font will be in enabled state, and all character codes will be checked for extension codes first. If you are entering codes other than 7-bit ASCII (such as ALT-248 for °), they might be taken as the lead code of a two-byte or multi-byte code and so the extension text style will be searched for the fonts. To use the full 8-bit IBM-extension character fonts and to avoid the mistake in code interpretation, enclose the string with a single '~', such that they are masked from the extension code checking.

To generate the font of '~', enter two successive '~'s to the text.

'{' **Left brace**, request to switch the text style from normal writing to superscript writing. The mid-point of the text height will become the start point position of the superscript writing. The text height is also scaled down as required (the system variable **TXTSUBFAC** is used to control the scaling factor). This function is also called the **Shift-In** function.

To generate the font of '{', enter two successive '{'s to the text. Note that this shift-in function may not be nested. If a '{' code is encountered when this shift-in function is in effect, the '{' code will be taken as an ordinary '{' code and will be generated.

'^' **Caret**, request to generate a line feed and carriage return, so that the next character writing will be at the first character position of the next line. If the writing is currently in Shift-In mode, the text superscript will be effectively changed to text subscript mode. The function is also called the **New-Line** function.

To generate the font of '^', enter two successive '^'s to the text.

'}' **Right brace**, request to return to the normal writing. The text starting position is restored back, and the pen position will be automatically aligned. This function is also called the **Shift-Out** function.

To generate the font of '}', enter two successive '}'s to the text. Note that if a '}' code is encountered when the shift-in function is not in effect, the '}' code will be taken as an ordinary '}' code and will be generated.

12H **Ctrl/R or ALT-18**, specifies to generate a vertical bar. The generation of this Vertical Bar function depends on the current active state of special control functions, in the following priority:

1. If both Overscore and Underscore functions are active, the vertical bar connects to both lines.
2. If Small Text Surrounding Box (**Ctrl/S**) function is active, the vertical bar connects to the box.
3. If Large Text Surrounding Box (**Ctrl/W**) function is active, the vertical bar connects to the box.
4. Otherwise, the vertical bar is generated from the Underscore position to the Overscore position.

14H **Ctrl/S or ALT-20**, starts/stops the generation of the **Small Text Surrounding Box (STSB)** function. The height of this text box is 1.75 times the text height.

17H **Ctrl/W or ALT-23**, starts or stops the generation of the **Large Text Surrounding Box (LTSB)** function. The height of this Large Text Surrounding Box is 2 times the text height.

18H **Ctrl/X or ALT-24**, starts or stops the generation of **Overscore** function.

19H **Ctrl/Y or ALT-25**, starts or stops the generation of **Underscore** function.

1DH **Ctrl/] or ALT-29**, starts or stops the generation of **Double Strike** function.

## Rules About Feature Functions

The start/stop control of a feature function in the text generation are governed by the following rules:

- Each function of the control codes are independent from each other. Cross-interleaving of different control codes are allowed. However, unreasonable arrangement of these codes in the text string may result in undesired effect on the text generation.
- Each of the function control codes is expected to appear in pair, one to initiate the feature function (indicating where to start) and one to terminate it. However, at the end of a line (logically by '^' or physically by the end of the string), all active feature functions will be terminated as a matter of consequence. The termination order is irrelevant to their appearing order. The **Overscore**, **Underscore** and **Double Strike** feature function are terminated first, and then the **STSB** feature function, and finally the **LTSB** feature function.
- The **Shift-In** function '{' (to open sub-line) will push the current status of these above functions in stack and take a fresh new start for the sub-line. And, the **Shift-Out** function '}' (to close sub-line) will restore the status of these functions before the opening of the sub-line. This makes the texts of tolerance (superscript/subscript) have their own individual controls, while the individual control shall not affect the overall controls.
- The lines generated by these feature function codes will not be affected by the **FILL** command.

Note: The codes to generate the texts as in the example figure left are as below:

```
<1Dh>Double Strike Text
<19h>Underline<18h> & <19h>Overscore<18h>
<17h><14h>box<14h><17h>
<14h>1{<19h>1 ^16}"<14h>
<17h><esc>h<12h>0.001<17h>
```

~~Double Strike Text~~

Underline&Overscore

box 1<sup>1</sup>/<sub>16</sub>" ⊥ 0.001

## Built-in Symbols

Built-in symbols are predefined text fonts that can be directly referenced in any text string. They are generated as a part of the text string regardless of the text styles used for the string. They are provided to facilitate the generation of special symbol used in dimensioning.

Built-in symbols are referenced by control codes or escape code sequences. Escape code sequences are code sequences introduced by the ESC code. They are described below:

<b>10H</b>	Ctrl/P or ALT-16, Taper Right symbol (CNS).	
<b>11h</b>	Ctrl/Q or ALT-17, Taper Left symbol (CNS).	
<b>16h</b>	Ctrl/V or ALT-22, Square Material symbol (CNS).	
<b>1EH</b>	Ctrl/^ or ALT-30, Single Triangle symbol	
<b>1FH</b>	Ctrl/_ or ALT-31, Upside-down Single Triangle symbol	

<ESC> a	CNS/ISO Symbol for Straightness	
<ESC> b	CNS/ISO Symbol for Flatness	
<ESC> c	CNS/ISO Symbol for Circularity	
<ESC> d	CNS/ISO Symbol for Cylindricity	
<ESC> e	CNS/ISO Symbol for Profile-of-any-line	
<ESC> f	CNS/ISO Symbol for Profile-of-any-surface	
<ESC> g	CNS/ISO Symbol for Parallelism	
<ESC> h	CNS/ISO Symbol for Perpendicularity	
<ESC> i	CNS/ISO Symbol for Angularity	
<ESC> j	CNS/ISO Symbol for Position	
<ESC> k	CNS/ISO Symbol for Concentricity and coaxiality	
<ESC> l	CNS/ISO Symbol for Symmetry	
<ESC> m	CNS/ISO Symbol for Maximum Object State (M in circle)	
<ESC> n	CNS/ISO Symbol for Circular Run-Out	
<ESC> o	CNS/ISO Symbol for Total Run-Out	
<ESC> p	CNS/ISO Symbol for Projected Tolerance Zone	
<ESC> <^P>	GB Symbol for Left Taper (^P means <b>Ctrl/P</b> , 10H, ALT-16)	
<ESC> <^Q>	GB Symbol for Right Taper (^Q means <b>Ctrl/Q</b> , 11H, ALT-17)	
<ESC> 0	GB Symbol for Right open angle with flat bottom surface	
<ESC> 1	GB Symbol for Left open angle with flat bottom surface	
<ESC> 2	GB Symbol for Right open angle with flat top surface	
<ESC> 3	GB Symbol for Left open angle with flat top surface	
<ESC> 4	CNS Symbol for Taper.	
<ESC> H	CNS/ISO Symbol for hexagon shape material	

All other escape code sequences are taken as undefined codes in text generation. You may enter the <ESC> code by pressing <Ctrl/[> or the ESC key, or **ALT-27**.

## System Background Control Codes

Some control codes are reserved for system usage in background control of the text in display. They are used for the screen display control purpose, and should not be entered as a part of the text string (you may confuse yourself if you try to enter these codes):

- 01H**     **Ctrl/A or ALT-1**, tells the **TCAM Graphic Runtime** to enable the interpretation of extended codes for subsequent characters in display. This code will not be displayed.
- 02H**     **Ctrl/B or ALT-2**, tells the **TCAM Graphic Runtime** to disable the interpreting of extended codes for subsequent characters in display. This code will not be displayed.
- 0FFH**    **ALT-255**, tells the **TCAM Graphic Runtime** to switch the display color attribute with the code following this 0FFh code.

### Procedure:

- **To write a text at an angle of 30°, 10 unit high:**
  - @CMD: **TEXT** (*return*)
  - Justify(L)/Style/<Start point>: (*point*)
  - Text Height <3.>: **10.** (*return*)

# TIFFOUT

## Export Drawing Images to Tagged Image Format File (TIFF) Command (Win)

---

### Purpose:

The **TIFFOUT** command is used to export the drawing image to disk file in TIFF (Tagged Image Format File). The image will be generated in the same way as the **PRPLOT** command prints the drawing to the printer.

### Description:

The **TIFFOUT** command lets you export drawing images to disk files in the following TIFF image types:

- Un-compressed monochrome image in white color background.
- Un-compressed monochrome image in black color background.
- PackBit compressed monochrome image in white color background.
- PackBit compressed monochrome image in black color background.
- Un-compressed 16-color image in white color background.
- PackBit compressed 16-color image in white color background.
- Un-compressed 256-color image in white color background.
- PackBit compressed 256-color image in white color background.

The images will be generated in the same way as you prints the drawing to the printer. However, apart from output to the printer, which has fixed resolutions and X/Y aspect ratio, printing image to bitmap file requires you to determine the output resolutions and the X/Y aspect ratio. So, the operation of the **TIFFOUT** command is about the same as the **PRPLOT** command, excepts that the details of the image export configuration is different.

In fact, **TIFFOUT** command works exactly the same way as **BMPOUT** command, excepts that the resulting bitmap image is converted and saved to the disk file in the required TIFF file format.

See **BMPOUT** command for the operation details and special notes about the image file export.

### Procedure:

@CMD: **TIFFOUT** (*return*)

What to plot -- Display, Extents, Limits or Window <D>:

...[Dialog Window Operation]...

### Example:

# 'TOOLBAR

## Toolbar Control Command.

---

### Purpose:

The **TOOLBAR** command is used to control the status of existing toolbars loaded from the current menu file in use.

### Description:

The **TOOLBAR** command lets you list out the status of all toolbars loaded from the current menu file in use and show up or hide out selected toolbars. You can select the toolbars by their names, which are defined in the menu file, or by an index number given in creation order.

When you issue **TOOLBAR** command, **TwinCAD** will prompt

Toolbar names (ALL, Index, or ? for a listing) <?>:

asking you to specify the name or the index number of the toolbars to select. If more than one toolbar are to be selected, their names or index numbers must be entered on the same input line and delimited with a single comma character.

You may enter '?' code or press space bar directly to the prompt to generate a list of all the toolbars defined in the system. An example listing is given below:

(B)	1:	Osnap	Object Snap
(B)	2:	Pucs	Perspective Operations
(B)	3:	Tcl	TCL Commands
(L)	4:	Slides	Slide Show Operation
( )	5:	Utility	Utilities
(B)	6:	System	System Operation
(R)	7:	Display	Display Control
(T)	8:	Dimension	Dimension Commands
(*)	9:	Edit	Editing Tool
(L)	10:	Entity	Entities Creation
( )	11:	Block	Block
(T)	12:	Standard	Standard Tools

The character code enclosed in the parenthesis indicates the docking status of the toolbar. The characters 'L', 'R', 'T' and 'B' means that the toolbar is docked to the Left/Right/Top/Bottom of the main frame window, respectively. The character '\*' means that the toolbar is floating, not docked to any side of the frame window. If the character is a blank character, it means the toolbar is currently hidden.

After you have selected one or more of the toolbars, it prompts

Hide/<Show>:

for you to determine their current status. You may enter the sub-command option "**H**" to hide out all the selected toolbar, or press the space bar to show up all of them. After that, the command ends.

Note that the docking status of a specific toolbar can only be changed via mouse pointer activity interactively. Also, you may directly hide out a specific toolbar at floating state by clicking at its tiny [-] button located at its top-left corner.

**Procedure:**

@CMD: **TOOLBAR** (*return*)

Toolbar names (ALL, Index, or ? for a listing) <?>:

Hide/<Show>:

**Example:**

# TRIM

## Trim Off Unwanted Objects Command

---

### Purpose:

The **TRIM** command is used to trim off the unwanted portion of objects in the drawing.

### Description:

The **TRIM** command lets you trim off the unwanted portion of objects in a drawing. An object may be divided into several portions by other objects that intersect with it. These objects are said to be cutting edges over the object. You may use this command to remove the unwanted portions defined by these cutting edges.

So, you will be asked to specify all the objects that serve as the cutting edges via the object selection operation. Lines, Arcs, Circles, Ellipses and Polylines are all valid entities for this purpose. See **SELECT** command for the object selection operation.

After the selection of cutting edges, you may start the trimming by picking the objects at the parts to be trimmed off. You trim the objects until you answer the prompt with a null return which ends the command. You may enter the sub-command option "**U**" to undo the last trimming.

You may also choose to trim off a set of objects crossed by specific lines by entering the sub-command option "**C**" to the prompt. You will be asked to drag a series of lines to cross over the objects to be trimmed off. All the objects crossed by the lines will be trimmed as if it were picked up at the point of the intersection with the crossing lines. This is an easy way for mass trimming operation.

Note that a circle or a closed polyline must be intersected at least twice to separate it into portions. You can't trim a circle or closed polyline with only one point of intersection. Also, only Lines, Arcs, Ellipse, Elliptic-arcs and Polylines can be trimmed off. Note also that an object is trimmed by the point of intersection with another cutting edge, as long as they do intersect in the model space. There is no restriction on whether they are parallel to the current UCS or not.

### Special Note

If a boundary object is trimmed off, it will no longer be a boundary object, while the remaining parts of it will be. That is, the trimmed-off portion of the object (which is erased from the screen) will not be an effective boundary for the subsequent trimming operation, while the remaining parts will remain highlighted on the screen and serve as effective boundaries.

Also, a boundary object with an extrusion thickness will be taken as a wall (not a single wire) used in the boundaries.

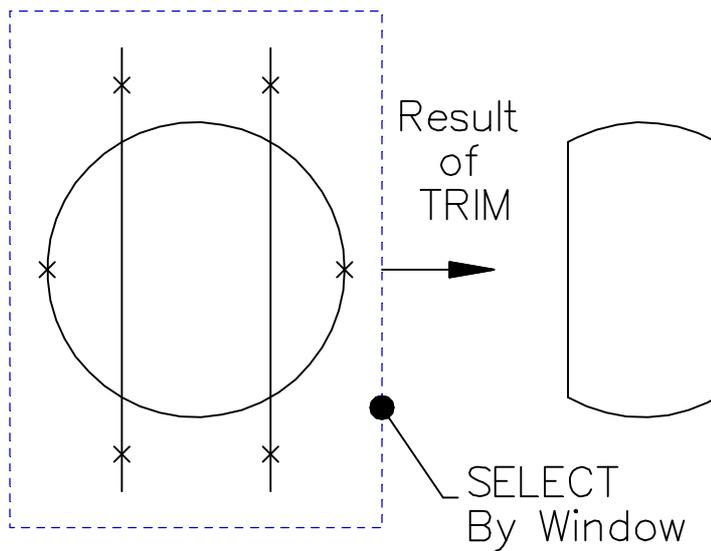
### Update:

For TwinCAD version after V3.1.048, the Text and Block Instance (Insert) entities can also be selected as the boundary objects. The virtual bounding boxes of the texts will be used as the boundary, and the block definitions will be traveled for valid boundary objects.

### Procedure:

To trim objects, follow the procedure below:

@CMD: **TRIM** (*return*)  
Select cutting edges....  
Select Objects (+): (*do so*)  
Undo/Cross-line/<Select object to trim>: (*pick*)  
...  
Undo/Cross-line/<Select object to trim>: (*return*)

**Example:**

**'UDFVAR**

## View Undefined Variable List Command

---

**Purpose:**

The **UDFVAR** command is used to view the list of undefined variables.

**Description:**

The **UDFVAR** command lets you view the list of undefined variables, which is created and maintained by **TwinCAD**. Most of the undefined variables are resulted from the loading of drawing file containing symbols and text style references that were not properly loaded.

For example, if a specific symbol in loading request can not be located in the symbol libraries found from the paths, then the name of that symbol will become an undefined variable and is put into the Undefined Variable List. Since the specific symbol was not loaded, all symbol references to it will not be drawn either.

Though the drawing loading command will give a report for the undefined symbols, you may forget to make the remedy later on. Therefore, using the **UDFVAR** command is a convenient way for you to keep track of the undefined symbols and take timely measures to do the remedy.

**Procedure:**

To view the Undefined Variable List, type as below:

@CMD: **UDFVAR** *(return)*  
*(Operate on pop-up window)*

**Example:**

## Restore the Previous State of CAD Database

---

### Purpose:

The **UNDO** command is used to restore the drawing database to the state before the last command execution that affected the database.

### Description:

The **UNDO** command lets you restore the state of the drawing database back to the one before the last restorable command which affected the database.

A restorable command is a command that affects the drawing database or system variables. For example, **DXFIN** command is a restorable command while **DXFOUT** is not. Therefore the **DXFIN** command can be "undone". In some cases, however, a restorable command may not necessarily be undone if it did not change the database. An example to this is the **ZOOMing/PANning** of view window, which is restorable (via 'ZP' command) but does not change the drawing database, and is therefore not affected by **UNDO** command. See **ZOOM** command.

When you enter the **UNDO** command, **TwinCAD** will prompt

Undo -- Control/<Number>:

You may input a positive non-zero integer value to specify the number of restorable command operations to be undone. This is an effective way to do multiple undo commands at a time. If you press space bar to the prompt, **TwinCAD** will take it as an input of 1, and will undo only once. This is equivalent to pressing the **UNDO** key.

**Procedure:**

To undo the last 5 commands, type as below:

@CMD: **UNDO** (*return*)

Undo -- Control/<Number>: **5**

**UNDUP**

## Check and Remove Duplicated Entities Command (TCL)

---

**Purpose:**

The **UNDUP** command is used to check and remove duplicated entities automatically from a set of selected entities.

**Description:**

The **UNDUP** command lets you remove the duplicated entities from the entities you select via the general object selection.

Once the **UNDUP** command is entered, it will prompt

```
UnDup -- Please select entities for duplication check...
```

and enter the general object selection operation. After you have completed the object selection, **UNDUP** will start checking to see if there are duplicated entities in the selection set. If there are, **UNDUP** will erase them and report the message:

```
Total nn duplicated entities have been removed.
```

Otherwise, it will report

```
No duplicated entities are found.
```

**UNDUP** will check only the geometry property of the entities, excluding the entity properties, such as layer, color and linetype.

**Limitation**

For speed consideration, the **UNDUP** will not process more than 4000 entities at a time. However, you may execute **UNDUP** several time in different area of the drawing to verify that there is no more duplicated entity.

**Special Notes:**

The **UNDUP** command is an external command provided by the TCL program file "UNDUP.TCL" or "UNDUP.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **UNDUP** command, you may solve the problem by copying the "UNDUP.TCL" to the COMMANDS sub-directory.

**Procedure:**

```
@CMD: UNDUP (return)
UnDup -- Please select entities for duplication check...
Select Objects (+): (do so)
...
```

**Example:**

**UNFILL****Remove the Solid Color Fill State of an Object Command**

---

**Purpose:**

The **UNFILL** command is used to remove the solid color fill state of selected objects.

**Description:**

The **UNFILL** command lets you remove the solid color fill state, imposed by the **FILL** command previously, of selected objects. See **FILL**command.

You will be asked to select objects to **UNFILL**.

**Procedure:**

To unfill the filled objects, type as below:

@CMD: **UNFILL** *(return)*

Select region, circle, closed polyline (+): *(do so)*

**Example:**

**'USERVAR**

## Access User-Defined Variable Command

---

**Purpose:**

The **USERVAR** command is used to access the current defined user variables.

**Description:**

The **USERVAR** command lets you access the current defined user variables in the drawing in the same way as **SYSVAR** command. See also **SYSVAR** for details about the operation.

User variables are created by the call to **TCL** runtime function **uservar()** for application purpose. A user variable, once being created, can be accessed in the same way as a system variable. It may be saved with the drawing to disk file, depending on its property being assigned. See ***TCL Command Language Reference Manual*** for details.

**Procedure:**

To access the current defined user variables, type as below:

@CMD: **USERVAR** *(return)*

*(Operate on pop-up window)*

**Example:**

# 'VARLIST

## List All Variables Command

---

### Purpose:

The **VARLIST** command is used to list out the current content of all the system variables.

### Description:

The **VARLIST** command lets you list out the current content of all the system variables to the command area. This command is provided mainly for diagnostic purpose. It is used to generate the listing so that it may be included with the error report if necessary.

### Procedure:

Enter the **VARLIST** command to **TwinCAD** command prompt:

```
@CMD: VARLIST (return)
```

and a listing of system variable in alphabetical order will be generated.

### Example:



## Vertical Linear Dimension Command

---

### Purpose:

The **VDIM** command is used to create a linear dimension in vertical direction.

### Description:

The **VDIM** command lets you generate a linear dimensioning which is always in vertical direction by

- **Two points**, of which the vertical projection distance is dimensioned, specifying the extension line origins. If this vertical projection distance is zero, **TwinCAD** will make a beep and ignore the dimensioning request.
- **A line**, of which the length projecting on Y-axis is dimensioned, and of which the two end points are used as the extension line origins. If the line selected is a horizontal line, **TwinCAD** will make a beep and ignore the dimensioning request.
- **An arc**, of which the chord length between two end points projecting on the Y-axis is dimensioned vertically. Again, if the length of this projection is zero, **TwinCAD** will make a beep and ignore the dimensioning request.
- **A circle**, of which the diameter is dimensioned vertically. The default dimension text will be generated as that for the Diametric dimensioning.

You will be asked to designate the first dimension point, and then the second dimension point. If you give the first prompt a null reply, you will then be asked to pick up object (Line/Arc/Circle) to dimension.

The dimension text is generated automatically in the drawing unit to reflect the true length of the object. Dragging of the dimensioning result is provided and is started for you to settle its placement, as soon as you finish the basic specification of the dimensioning.

There are several ways to settle the placement of dimension text as well as the dimension line. During the dragging of dimensioning, you may

- **Enter a distance value**, specifying the distance from the dimension line to the first dimension point, where the dimension text position is determined automatically by the current dragged position when the value is entered. Note that the dragged position determines only the projected position of the text to the dimension line.
- **Enter nothing but space bar**, specifying a default distance for the dimension line should be used. The determination of a default distance depends on the mode of dimensioning. See later explanation.
- **Designate a point**, specifying directly where the dimension text should be placed at. The dimension line will be generated automatically according to the setting of dimension variables (specifically, **DIMGAP**). You may use the cursor snap function (in small step) to help you locate the desired dimension line position, if casual alignment with other dimension lines is intended.

Before settling down the placement of the dimensioning, you may modify the dimensioning by the following sub-command options:

**D** ***Default text***, specifying to reset the dimension text to the system default. The default text is generated in association with the current measurement of dimension and the setting of dimension variables. You may access these variables by issuing the **DIMVAR** command. See also **DIMVAR** for more information about it.

**C** ***Change text***, specifying to change the dimension text manually. **TwinCAD** will prompt:

Dimension Text:

and open a text entry field for you to modify the existing dimension text. Note that once the dimension text is entered in this manner, it will be fixed regardless of the possible change of the dimension measurement thereafter, until it is reset by the Default-text sub-command option.

You may effectively disable the text generation by changing the text to '~' only.

**OD** ***Dimension-Line Option***, specifying to toggle the current effect of dimension line option. The dimension line option is effective only when the dimension text is dragged outside of the dimension points and so are the dimension lines generated outside of the extension lines of the linear dimensioning. When this option is ON, an additional straight line will be added inside of the extension lines, connecting the two dimension lines which fall outside of the extension lines. This option can be preset at the bit 0 of the dimension variable **DIMLOPT**.

**OT** ***Text generation option***, specifying to toggle the current effect of text generation option. It specifies whether to align the dimension text with the dimension line or not. If this option is OFF, the dimension text generated will always be horizontal. But if it is ON, the dimension text will be generated in the same direction as the dimension line. This option can be preset at bit 1 of the dimension variable **DIMLOPT**.

**OA** ***Text alignment option***, specifying to toggle the current effect of text alignment option. It specifies whether to align the dimension text above the dimension line or not. If this option is ON, the dimension text will be placed at a position such that the dimension line leads to the middle height of the text. The dimension line will be clipped if necessary. If it is OFF, the text will be placed above the dimension line (and the dimension line will be extended to cover the span of text if necessary). This option can be preset at bit 2 of the dimension variable **DIMLOPT**.

**OR** ***Dimension-Line Reverse Option***, specifying to toggle the current effect of dimension line reverse option, which means to reverse the dimension line with respect to the dimension text. If this option is ON, the dimension line and arrows will be placed in the reverse direction. That is, if the text is inside, the arrows will be outside pointing inward. If the text is outside, the dimension line will be inside the extension lines. This option can be preset at the bit 3 of the dimension variable **DIMLOPT**.

**A** ***Align Option***, specifying to settle the dimension line placement, such that it can be fixed to align with a specific point or at a specific dimension distance. **TwinCAD** will prompt:

Indicate a point to align the dimension line:

asking you to designate the alignment point. You may directly pick at an existing dimension to align with it, as **TwinCAD** will automatically issue the **ENDp** snap directive, or enter a specific distance value, to fix the dimension line position. Once the dimension line position is fixed, only the dimension text position can be dragged and changed by the cursor pointer.

You may release the fixed dimension line by entering this option again and pressing space bar to the prompt.

After the first dimensioning is done, **TwinCAD** will continue to prompt for the same dimensioning operation with the following added options:

- U**     **Undo**, request to undo the last dimensioning.
- B**     **Base**, request to generate Base Dimensioning, using the first extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimension set.
- When the Base dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Base dimension mode may be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.
- C**     **Continuous**, request to generate Continuous Dimensioning, using the second extension line origin of the last dimensioning as the first extension line origin for the next dimensioning. You will be asked to input the second dimension point, since the first one is determined by the last dimensioning.
- When the Continuous dimension mode is entered, the oblique angle assumed by last dimension set will be retained and become a default for subsequent dimension sets. The Continuous dimension mode may also be automatically invoked when you re-enter the dimension command and pick up one end point from an existing linear dimension line as the first dimension point.
- Note that once the Continuous or Base dimension mode is entered, it will be continued automatically after each dimensioning is done. However, you may press the space bar to the prompt asking for the second dimension point to quit the mode and to go back to the first prompt where you may have other options as usual.
- H**     **Horizontal**, request to switch to horizontal linear dimensioning as if you have exited the operation and enter the **HDIM** command.
- V**     **Vertical**, request to switch to vertical linear dimensioning which is meaningless in this context as you are doing vertical linear dimensioning.
- A**     **Aligned**, request to switch to aligned linear dimensioning as if you have exited the operation and enter the **ALDIM** command.

The last three sub-command options from the above are added for easy mode switching between the Vertical, Horizontal and Aligned Linear Dimensioning. You may identify which mode you are in now by checking the parenthesized character (**H**, **V** and **A**, respectively) from the command prompt.

You have to type **<Ctrl/C>** to exit the command operation or give a null reply when you are asked to pick up object to dimension.

## Default Dimension Line Distance

The determination of default distance of dimension line follows the rules below:

- If Base dimension mode is selected, the default distance will be the last dimension line distance plus or minus the default dimension line interval, depending on whether the second dimension point is outside of the extension lines of the last dimension or not.
- If Continuous dimension mode is selected, the default distance will be determined such that the dimension line of the newly created one will be on the same vertical level with that of the last one.
- If none of the above situations happens, the default distance will be the default dimension line interval specified by the dimension variable **DIMDLIR** (see **DIMVAR** for details).

## Suppression of Extension Lines

You may use the dimension variables **DIMSE1** and **DIMSE2** to specify the default suppression status of dimension extension lines. However, it is advised not to do so, since it will slow down your productivity. It is better to use the **SETDIM** command to adjust these items after all the dimensioning jobs are done. Do not get annoyed with these variables during the dimensioning operations. They are provided for other purposes, such as dimensions generated by TCL programs. The **SETDIM** command is so designed as to make you feel free with the dimensioning operations.

However, for Base Dimensioning and Continuous Dimensioning, it is obvious that some extension lines would overlap as they are generated. It is therefore necessary to suppress one of them (the shorter one, of course). Sophisticated check is performed automatically to see which one should be suppressed when the dimensioning is done under these two modes. The effects of the **DIMSE1** and **DIMSE2** are ignored in these two modes.

## Virtual Segments Extension

The virtual segments generated by Dimension or Symbol entities can be accepted as valid object selections during Dimension operations. For example, you may select the extension lines and put on dimensions using the **VDIM** command.

## Dimensioning on PUCS-Plane

The **VDIM** command is able to recognize the current PUCS-plane setup of an Axonometric Projection, and allows you to produce dimensions on the projected plane from the virtual space directly.

If the current PUCS-plane is active and valid for an Axonometric Projection, the dimension points will be measured from the Projected UCS-plane, and the **VDIM** command will automatically calculate the required Oblique angle and set up the Rotated direction in the X-axis direction for the dimensioning as a projection result of the vertical linear dimensioning from the Projected UCS-plane to the model space (as the same transformation result done by **MVCOPY** command).

The Ellipse and Elliptic-arcs that can be produced by using current Axonometric Projection setup can be picked up for the linear dimensioning. The result of the dimensioning will be the same as the dimensioning of a circle or an arc on the projected plane from the virtual space.

The **VDIM** is actually a Rotated **ALDIMs** with zero oblique angle, since their extension lines are always fixed to the X-axis (horizontal) direction.

### Procedure:

Enter the **VDIM** command to **TwinCAD** command prompt:

@CMD: **VDIM** (*return*)

and **TwinCAD** will prompt

Horizontal/Vertical/Aligned/(V)<First dimension point>:

If you reply to it by designating a point, you are dimensioning by two points. If you reply to it by pressing the space bar, **TwinCAD** will let you dimension by selecting objects. See the procedure for each case below:

- **Two points:**

Horizontal/Vertical/Aligned/(V)<First dimension point>: (*point*)

Second dimension point: (*point*)

Default-text/Change:<text>/OD(Dimension-line option)/  
<Set dimension line position or distance>: *(Drag to position)*

- **A line, arc or circle:**

Horizontal/Vertical/Aligned/(V)<First dimension point>: *(space bar)*

Select arc, line or circle: *(pick up one)*

Default-text/Change:<text>/OD(Dimension-line option)/

<Set dimension line position or distance>: *(Drag to position)*

After the first dimensioning is done, **TwinCAD** prompts

Horizontal/Vertical/Aligned/Base/Continue /Undo/(V)<First point>:  
to continue the operation.

**Example:**

# 'VIEWOPT

## Screen Drawing View Options Setup Command(TCL)

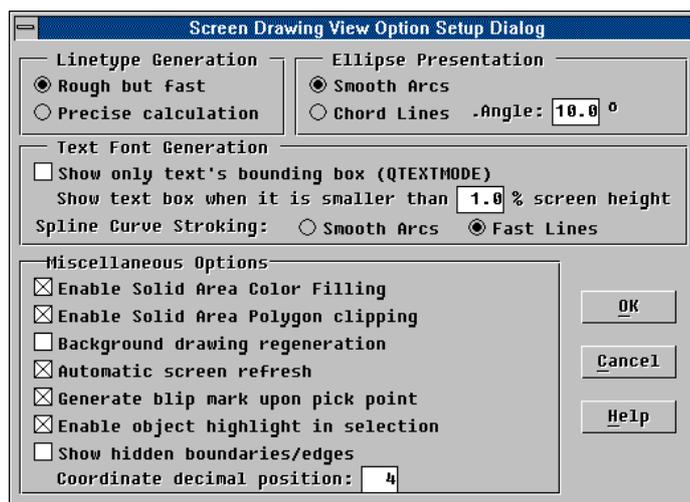
### Purpose:

The **VIEWOPT** command is used to setup screen drawing view related options via a GUI dialogue operation.

### Description:

The **VIEWOPT** command lets you access the screen drawing view related options via a GUI dialogue window operation. These options, such as the fast linetype generation, smooth ellipse representation, quick text mode control and fast text curve stroking, are controlled by specific system variables. With **VIEWOPT**, you can easily access these options without knowing how to setup these related system variables.

**VIEWOPT** will pop up the dialogue window as shown below:



The following describes these screen items in details.

### Linetype Generation Group

There are two ways to generate a screen linetype.

**Rough but fast** The pattern linetype will be generated in graphic driver level, which is fast but not very precise to the definition of the pattern linetype. The pattern linetype will not be adjusted for both end points of a line.

**Precise calculation** The pattern linetype will be generated with precise calculation such that the screen view will be exactly as what is expected from the printer/plotter output.

Note that this option is effective to screen linetype only. It will not affect the printer/plotter output.

## Ellipse Generation Group

There are two ways to generate an ellipse on the screen.

- Smooth Arcs**      The ellipse will be drawn with successive smooth arc segments that approximate the ellipse to a very high precision. For a whole ellipse, this approximation will take up 32 arcs.
- Chord Lines**      The ellipse will be drawn with successive chord line segments. It is faster but may have undesired polygon effect if the stepping chord angle is too large. You may control the fineness of the ellipse by specifying a proper chord angle.

Note that this option is effective to the ellipse drawn in the screen. It will not affect the printer/plotter output.

## Text Font Generation Group

You may choose to display only the bounding boxes of texts in the screen without actually drawing the text stroke details, to faster up the drawing regeneration during your drawing editing session. Or, if you prefer, you may specify a threshold control value for **TwinCAD** to decide which texts will be drawn with stroke details and which will not. This threshold control value is defined as the ratio of the text size in height to the height of the drawing window, given in percentage. All texts that are smaller than this threshold value, will be drawn with bounding boxes only.

Text fonts from TrueType or PFB files may contain Bezier curve elements. **TwinCAD** will realize these elements with smooth arc segments when the text fonts are drawn, plotted or exploded. However, if you are using a lot of TrueType fonts or Adobe's PFB fonts in the drawing, you may like to speed up the drawing view regeneration by specifying to draw the Bezier curve elements using fixed line segments. Note that this option will not affect printer/plotter output, nor the result of text in explosion.

## Miscellaneous Options Group

There are other interested options in this group.

### Enable Solid Area Color Filling

If this option is enabled, all entities with solid color fill attribute (such as Circle, close Polyline, Region, Text) or with wide line property will be actually filled with specified solid color in the screen. Otherwise, only profiles of these solid areas are drawn. You may turn off this option to speed up the drawing regeneration.

### Enable Solid Area Polygon clipping

If this option is enabled, a solid area of color fill will be clipped with the virtual drawing window before it is passed to the graphic driver for color rendering. If you turn off this option, **TwinCAD** will temporarily turn off the solid color filling if one should be not totally within the virtual drawing window, which will speed up drawing regeneration.

### Background Drawing Regeneration

If this option is enabled, **TwinCAD** will regenerate the drawing in the background bitmap first and then show the bitmap to the foreground window at once. Doing so would be faster in drawing regeneration, yet without the real time feeling.

### Automatic screen refresh

If this option is enabled, certain prime commands will issue screen refresh request to redraw the screen, so that these highlighted objects (on temporary basis) or pointer markers (blip mark) will be removed immediately from the display. This refreshing will cause **TwinCAD** to clear the graphic window and redraw the display list again. Usually, this is fast and latent to the operator.

However, if you turn this option off, these highlighted objects will be removed in another way, which may be faster but the blip marks will not be cleared.

This option is controlled by AUTOREDRAW variable.

### Generate blip mark upon pick point

If this option is enabled, **TwinCAD** will generate a point marker (in small crossline) right at the snapped position when you pick up an object or designate a point. The marker thus generated is not part of the drawing, yet helps you to identify the pickup operation. It will be removed when the screen is refresh by the **REDRAW** command.

This option is controlled by BLIPMODE variable.

### Enable object highlight in selection

If this option is enabled, all objects in selected state will be highlighted in the screen, so that they can be clearly identified by the operator. If this option is off, selected objects will not be highlighted.

This option is controlled by HIGHLIGHT variable and is intended for TCL application to fast up the command processing.

### Show hidden boundaries/edges

If this option is enabled, the drawing regeneration will also draw out hidden boundaries (associated hatch regions) and edges (3DFACE).

### Coordinate decimal position

This is a value entry, specifying the number of decimal position for the screen coordinate display. If it is zero, 3 decimal position will be used but the trailing zero will be removed.

## Other Buttons

After completing your option setup, you have to press the **[OK]** button to confirm the change and terminate **VIEWOPT**. Or, you may press the **[Cancel]** to quit **VIEWOPT** operation. You may also press <ESC> key or the right mouse button to quit the operation.

## Special Notes:

The **VIEWOPT** command is an external command provided by the TCL program file "VIEWOPT.TCL" or "VIEWOPT.TCA". Either file must be present in the COMMANDS sub-directory where the **TwinCAD** resides. If you can not issue **VIEWOPT** command, you may solve the problem by copying the "VIEWOPT.TCL" or "VIEWOPT.TCA" to the COMMANDS sub-directory.

**Procedure:**

Enter the **VIEWOPT** command to the system command prompt:

@CMD: **VIEWOPT**

...[Dialogue Operation]...

# 'VPOINT

## View Objects from 3D View Point Command

---

### Purpose:

The **VPOINT** command is used to generate a 3D-view of objects in current view window.

### Description:

The **VPOINT** command lets you select a 3D view point and view the drawing in the direction taken from the point toward the origin. It creates a parallel projection of the drawing on the display in the specific view direction. There is an axis tripod displayed at the lower left corner of the drawing window to reflect the current view angle of the parallel projection.

When you issue the **VPOINT** command, **TwinCAD** will prompt

Dynamic/Rotate/<View point><1.,1.,1.>:

asking you to specify the relative view point to the drawing origin. You specify this view point by entering the 3D coordinates from the keyboard explicitly. The angle direction from the specified 3D point to the drawing origin (0,0,0) will be taken as view direction of the parallel projection.

### View Point by Rotation Angle

You may specify the viewpoint in terms of two angles, one with respect to the X axis (in the X/Y plane) and the other toward the Z-axis away from the X-Y plane, by entering the sub-command option "R" to the prompt. **TwinCAD** will prompt in sequence:

Enter angle in X-Y plane from X axis <45.°>:

Enter angle from X-Y plane <35.264°>:

for the two angles. The equivalent view point will be calculated as  $(\text{Cos}(A), \text{Sin}(A), \text{Tan}(B))$ , where A is the angle of the vector projected on the X-Y plane, and B, the angle between the viewing direction from the X-Y plane.

### Dynamic View Point

If you do not have any particular angle in mind to view the objects, you will probably want to rotate the objects dynamically and decide which view angle is best for you. To do this, enter the sub-command option "D" to the prompt. **TwinCAD** will immediately let you rotate the viewpoint by the cursor movement while refreshing the display for you. The cursor keys on the keypad can also be used to rotate the dynamic view point, as if you were rotating the object to view.

The keyboard inputs that may affect the dynamic rotation of view point are described below:

- |                    |   |
|--------------------|---|
| <b>Left Arrow</b>  | Rotate the object around its Z-axis counter-clockwise by 10°. If the Scroll/Lock key is engaged, this key will pan the object to the left (view to the right) by about 1/25 of the window size. |
| <b>Right Arrow</b> | Rotate the object around its Z-axis clockwise by 10°. If the Scroll/Lock key is engaged, this key will pan the object to the right (view to the left) by about 1/25 of the window size.         |

<b>Up Arrow</b>	<p>Rotate the object around the horizontal axis on the view plane clockwise by 10°. The rotation will maintain the projected z-axis on the vertical direction. It seems that the object is rotated in front of you in the backward direction (rolling away from you).</p> <p>If the Scroll/Lock key is engaged, this key will pan the object upward (view to the bottom) by about 1/20 of the window size.</p>
<b>Down Arrow</b>	<p>Rotate the object around the horizontal axis on the view plane counter-clockwise by 10°. The rotation will maintain the projected z-axis on the vertical direction. It seems that the object is rotated in front of you by rolling toward you.</p> <p>If the Scroll/Lock key is engaged, this key will pan the object downward (view to the top) by about 1/20 of the window size.</p>
<b>PgUp</b>	A combination effect of Up/Right arrow keys.
<b>PgDn</b>	A combination effect of Down/Right arrow keys.
<b>Home key</b>	A combination effect of Up/Left arrow keys.
<b>End key</b>	A combination effect of Down/Left arrow keys.

To fix the view point during the dynamic rotation, you may press the return key or the first button of the mouse. You may also type **<Ctrl/C>** to exit the dynamic rotation. Note that the menu script will not be read during the dynamic rotation operation.

Note: You may temporarily pan the view window by keeping the keyboard in Scroll/Lock state during the dynamic rotation of view point. However, if you have panned the view window, you better regenerate the drawing after view direction is fixed, or the cursor movement on the display will also be panned by the graphic driver.

## The Center of View Point

The center of the view point (the point about which the object is rotated to view) is determined by the center of the current view window. However, the Z coordinate of this center point is specified by the system variable **VZCENTER**. There is also one system variable **VZSCALE** related to this 3D view generation. It controls the relative scale factor of the Z coordinate in the projection, and is normally set to be 1. You may use the **SYSVAR** command to access these two variables. See **SYSVAR** command.

Note that, in 3D-View, although you may use **Zoom/Window** to view a specific portion of the drawing in detail from the screen, the object window center (for **VPOINT** rotation center) may not be very close to the objects seen in the screen area. So, the **VPOINT** operation may change the view dramatically. In this case, you can't expect to rotate the view of that specific portion in the view window, since a small angle of rotation will rotate it out of the window. One way of doing it is to use the **ZOOM/Center** and pick up a Near Point on the object at about the center of the screen, before using the **VPOINT/Dynamic** command.

## Special Note

A view point of (0,0,1) will view the drawing from the Z-axis, as you view it from the normal 2D plan view. However, as the axis tripod is in display, it is definitely not the same view created by the normal 2D-Plan view because all the Z-axis information is ignored in the 2D-plan view.

**Procedure:**

- **To view the drawing in isometric view by giving view point, follow procedure below:**  
@CMD: **VPOINT** *(return)*  
Dynamic/Rotate/<View point><0.,0.,1.>: **1,1,1** *(return)*
- **To view the drawing in isometric view by rotation of angle, follow procedure below:**  
@CMD: **VPOINT** *(return)*  
Dynamic/Rotate/<View point><0.,0.,1.>: **R** *(return)*  
Enter angle in X-Y plane from X axis <0>: **45** *(return)*  
Enter angle from X-Y plane <90.>: **45** *(return)*
- **To view the drawing by dynamic rotation, follow procedure below:**  
@CMD: **VPOINT** *(return)*  
Dynamic/Rotate/<View point><0.,0.,1.>: **D** *(return)*  
*(Rotate the view point as you like)*

**Example:**

# VSLIDE

## View Slide File Command

---

### Purpose:

The **VSLIDE** command is used to view the vector image of a slide file.

### Description:

The **VSLIDE** command lets you view the vector image of a particular slide file produced by **MSLIDE** command. A slide file is a file containing a picture of view once in the graphic display. The **VSLIDE** command lets you view the picture again. It is used mainly for presentation purpose, especially in the command script. This is an ACAD-compatible command.

When you enter the **VSLIDE** command, the file window will be popped up for you to select the slide file to view.

Once a slide file is loaded to view, the current drawing area will be temporarily cleared and used to draw the vector image read from the file. Note that you can not edit such view nor to print or plot it out, since it is only a temporary picture of view. Once you redraw the screen, it will disappear and the original view of your drawing will be restored.

You may create slide libraries that contain many slide files by using the utility **ASLIDE** shipped with the package. However, you can not view these slides in a slide library interactively. You can only view them by the **VSLIDE** command in a command script. The format to specify a slide in the **VSLIDE** command from a command script is as below:

*library*(*slide-name*)

where

- **library** -- the name of the library file.
- **slide-name** -- the name of the slide within the library (original file name without path).

The file extension for a slide library file is always ".**SLB**".

The utility **VSLIDE.EXE**, which is also shipped with the package, can be used to view the slides or slides in libraries in an interactive way or using batch command files for presentation purpose, without entering **TwinCAD**. See the document of **TwinCAD Utilities**.

### Procedure:

@CMD: **VSLIDE** (*return*)

(*Select slide file from file window*)

### Example:

# WBLOCK

## Write Block of Drawing to Disk Command

---

### Purpose:

The **WBLOCK** command is used to write a block of drawing to disk.

### Description:

The **WBLOCK** command lets you write out selected objects or a specific block definition to a disk file as a complete drawing module. The file so written out can be loaded later on as a drawing file, or inserted via the **INSERT** command.

Unlike the **SAVE** command, which saves the whole drawing to disk, the **WBLOCK** command saves only selected part of it. However, those objects, such as **BLOCKS**, **LAYERS**, **LINETYPES**, **STYLES** and **SYMBOLS** which are referenced by the selected part of the drawing, are also saved to make the drawing module complete, this include those system variables.

When you issue the **WBLOCK** command, **TwinCAD** will prompt

```
Wblock -- Block/<Entities>:
```

If you enter the sub-command option "**B**", **TwinCAD** will pop up a slide window of block name selection for you to select the specific block definition to output. After that, the file window will be popped up for you to specify the output file name, and the block is output accordingly. The block definition is output as the top level of drawing (not a block definition of drawing).

If you reply to the prompt with a null return, you are going to write out selected entities. **TwinCAD** will enter the Object Selection Operation (see **SELECT** command) for you to select the objects to output. After that, you will be asked to designate the insert base point of those selected objects, as **TwinCAD** prompts:

```
Insert base point:
```

This insert base point is essential to the drawing module when it is referenced for insertion into other drawings. See also **BASE** command. Finally, a file window will be popped up for you to specify the output file name, and all the selected objects will be output accordingly.

Note that, the output file will be in DWG file format if the filename extension is explicitly given as "DWG"; otherwise, it is saved in WRK format.

### Procedure:

```
@CMD: WBLOCK (return)
Wblock -- Block/<Entities>: (return)
Select Objects (+): (do so)
Insert base point: (point)
File name<>: (Specify file in file window)
```

# WMFOUT

## Export Drawing Images in Windows Meta File Format Command

---

### Purpose:

The **WMFOUT** command is used to export the drawing image to disk file in Windows Meta File format (WMF). The image will be generated in the same way as the **PRPLOT** command prints to the printer.

### Description:

The **WMFOUT** command lets you export drawing images to disk files in the following WMF formats:

- Placeable Windows Meta File.
- Standard Windows Meta File.

The images will be generated in the same way as you prints the drawing to the printer. However, apart from output to the printer, which has fixed resolutions and X/Y aspect ratio, printing image to Windows Meta file requires you to determine the output resolutions and the X/Y aspect ratio. So, the operation of the **WMFOUT** command is about the same as the **PRPLOT** command, excepts that the details of the image export configuration is different.

In fact, **WMFOUT** command works exactly the same way as **BMPOUT** command, excepts that the resulting image is in different format. See **BMPOUT** command for the operation details.

### Special Notes

The contents of a WMF file is not bitmap, but a series of records of Windows GDI function calls. It is used in Windows system for graphic regeneration and related manipulations, just like the display lists to a CAD system.

The Standard WMF is the original WMF supported by Windows, while the Placeable WMF is a modified version for desktop publishing purpose. The latter has an additional 22 bytes header containing the image size information required by the desktop publishing software to determine the default frame size for the image at insertion.

Therefore, if you are creating WMF file for drawing insertion into a desktop publishing softwares like WinWord, you must use Placeable WMF format. Most of the desktop publishing softwares do not accept Standard WMF, because it does not contain the size information of the image. However, if you are to include a WMF file into a Windows application's resource, and draw it directly with Windows API PlayMetaFile() function call, you must export it in Standard WMF format.

### Procedure:

@CMD: **WMFOUT** (*return*)

What to plot -- Display, Extents, Limits or Window <D>:

...[Dialog Window Operation]...

### Example:

# WTCAM

## Optional CAM package for NC Wire-Cutting Machine

---

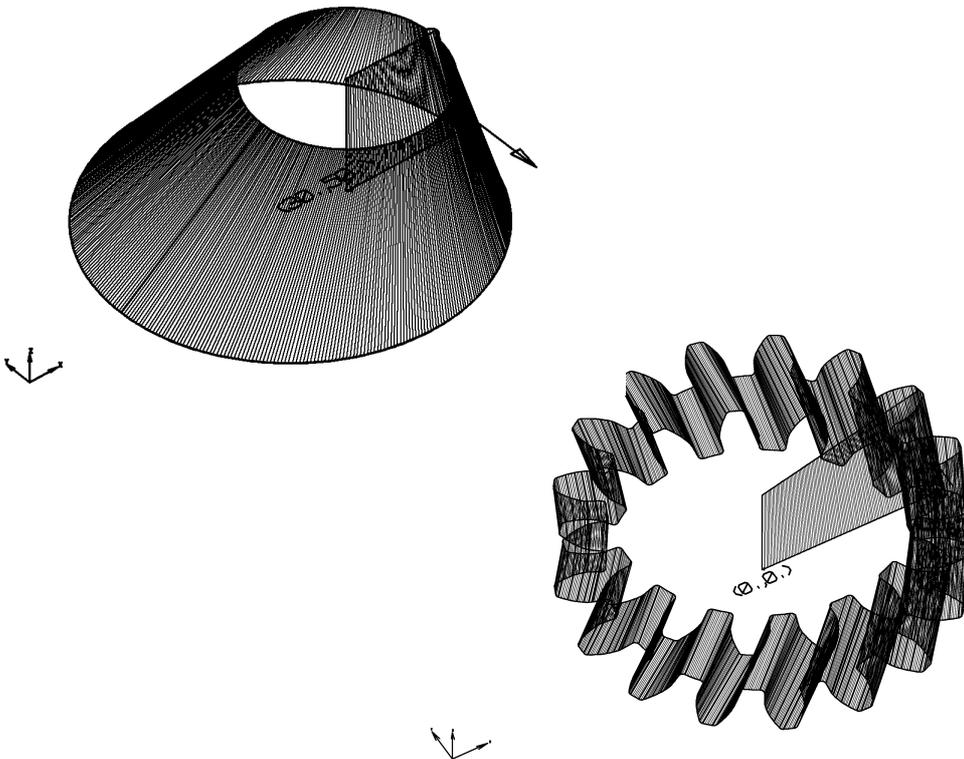
### Purpose:

The **WTCAM** command is used to access the optional CAM package for NC wire-cutting machine.

### Description:

The **WTCAM** command lets you prepare and make the tooling paths for NC wire-cutting machine and the likes. This is an optional package specific for NC wire-cutting machine. Detailed information regarding this command operation and other related technical information are given in a separate document.

Interested user may contact our local dealer for further information.





**XDIM**

## Create X-Ordinate Dimension Command

---

**Purpose:**

The **XDIM** command is used to create an X-Ordinate dimension.

**Description:**

The **XDIM** command lets you dimension designated points by their X coordinates relative to a given origin. The dimension lines are always in vertical direction.

When you issue the **XDIM** command, you will be asked to designate a point as the ordinate's relative origin for the subsequent dimensioning. After that, **TwinCAD** will continuously ask you to designate points to dimension and the locations to place the dimension texts. The dimension text for a designated point is automatically generated to reflect its true ordinate distance from the origin.

The placement of the dimension text can be automatic or by manual dragging, depending on current active option selected. The dimension line, starting from the designated point to the text, may be split into two parts which are joined by a horizontal lead line. This allows a more flexible way in placing the dimension text and will avoid possible overlap between two adjacent dimensions.

The following lists the sub-command options:

- L**      **Lead**, specifying that an additional lead line is required. If this option is active, the dimension text can be dragged in both X and Y directions, and the dimension line will be split into two parts in the middle with a horizontal lead line connecting both parts. After you have determined where to place the dimension text, **TwinCAD** will let you adjust the lead line location by dragging it vertically.
- N**      **No Lead**, specifying that no lead lines will be generated for the dimension lines. If this option is active, the dimension text can only be dragged in Y direction, and once its location is determined, the dimensioning is completed.
- D**      **Drag**, specifying that the text position is free to drag, not to be fixed.
- F**      **Fix**, specifying that the text position is fixed at the same Y coordinate level as the last dimensioning. In the case when the Lead option is active, you can still drag the dimension text but only in the X direction, as its Y coordinate has been fixed. In normal case, however, the dimension text is placed automatically as soon as you designate the point to dimension. Note that you must explicitly determine the dimension text position of the first dimensioning.
- FL**    **Fix in Length**, specifying that the text position will be at the Y coordinate level with a fixed vertical distance from the designated point. That is, the length of the dimension line will be fixed and is the same as that for the last dimensioning. In the case when the Lead option is active, you can still drag the dimension text but only in the X direction, as its Y coordinate has been determined. Normally, however, the dimension text is placed automatically as soon as you designate the point to dimension.
- U**      **Undo**, request to remove the last dimensioning.
- C**      **Change-text**, request to change the dimension text.
- X**      **X-ordinate** dimensioning, request to switch to the X-ordinate dimensioning, meaningless as you are already in X-ordinate dimensioning.

**Y** **Y-ordinate** dimensioning, request to switch to the Y-ordinate dimensioning.

**A** **Auto**, request to produce the ordinate dimensioning automatically.

To exit the command, reply to the prompt with a null return or type <Ctrl/C>.

## Automatic Ordinate Dimensioning

The interactive dragging function provided by **TwinCAD** to produce the ordinate dimensioning is quite convenient for you to do the job already; nevertheless, if the drawing to dimension is too complicated, it still takes you much time to determine the best placement of the dimension texts to avoid possible overlaps. Once in a while, you may want to edit these dimensions by the **CHANGE** command to adjust their placement of dimension texts.

**TwinCAD** provides the feature of Automatic Ordinate Dimensioning to overcome this bottleneck. Simply select the objects that contain points to dimension, and **TwinCAD** will automatically generate the ordinate dimensioning for you. The placements of these dimension texts are so adjusted that they will not overlap each other, and are distributed as evenly as possible, while keeping the need for additional lead lines at the minimum. Of course, points that result in redundancy are ignored in the dimensioning.

To invoke the auto-dimensioning feature, enter the sub-command option "**A**", and you will be asked to select objects to dimension. Valid objects are:

- **Point** -- The point is taken to dimension.
- **Line** -- Both end points are taken to dimension.
- **Arc** -- Both end points are taken to dimension.
- **Circle** -- The Center point is taken to dimension.

After that, **TwinCAD** will prompt

Generate dimension only on parallel segments ? <N>:

If you reply YES, then only those line segments which are parallel to the dimension line direction (i.e., vertical for **XDIM**, horizontal for **YDIM**), circles and points are taken to dimension. The default is to take all the objects selected.

After that, if the dimension text position is not yet fixed (done by sub-command option "**F**"), you will be asked to designate a point with which the placement of dimension texts must align, with the prompt:

Please Indicate text position or length of lead line:

If you enter a value instead of a point to the prompt, then the length of dimension line will be fixed (as done by sub-command option "**FL**") at the given value.

Finally, **TwinCAD** prompts

*nn* generated. Use CHANGE command to adjust possible conflicts.

to report that the job is done.

## Procedure:

- **To generate the X-ordinate dimension, follow the procedure** below:
  - @CMD: **XDIM** (*return*)
  - Indicate original point <0.,0.>: (*point*)
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: (*point*)

Lead/Nonlead/FLen/Fix/Drag/Undo/<text position>: (point)

To fix the dimensioning on the same level

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: **F** (return)

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: (point)

To undo the last dimensioning

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: **U** (return)

To dimension with a lead line

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: **L** (return)

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: (point)

Lead/Nonlead/FLen/Fix/Drag/Undo/<text position>: (point)

Settle lead lines location: (point)

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: (point)

...

- **To generate dimension automatically**

Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(X)<dimension point>: **A**(return)

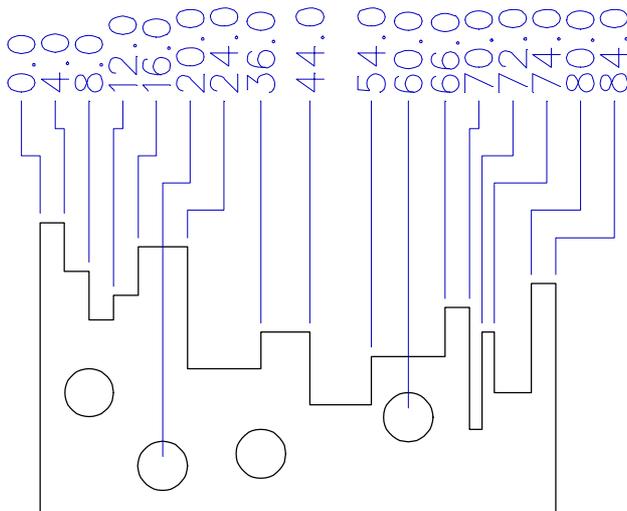
Select Objects (+): (do so)

Generate dimension only on parallel segments ? <N>: (as you wish)

Please Indicate text position or length of lead line: (point)

nn generated. Use CHANGE command to adjust for possible conflicts.

**Example:**



Ordinate Dimensioning Done Automatically

**XPDL**

## Explode Part of Dimension to Line Command (TCL)

---

**Purpose:**

The **XPDL** command is used to replace the extension line of a dimension with a real line for further trimming or breaking purpose.

**Description:**

The **XPDL** command lets you easily replace the indicated extension line of a dimension with a real Line entity that can be trimmed or broken into parts. It is used when you need to trim or to make a break on a dimension by its extension line so as to escape it from other drawing entities that intersect with it.

To make a trimming or breaking on the extension line of a dimension entity, most people would instantly explode the dimension into elementary entities first and then do the editing thereafter. However, this is not the recommended way, because it is too wasteful in terms of drawing resource. The exploded result can never be a dimensioning again to **TwinCAD** or to any other applications that read it.

In fact, if you are going to make the extension line shorter, you may apply the **QCHANGE** command directly. If you are going to separate the extension line into parts, you may turn it off via the **SETDIM** command, create an ordinary line entity as a replacement via the **LINE** command, and then do the required editing thereafter. The dimension entity is thus retained and only one additional line entities is created. And, that is what **XPDL** does for you.

As most of the purpose of this kind of editing is to escape the extension line from some other entities at certain points by making breaks of it, the **XPDL** will also create the breaks for you if you require.

**XPDL** will prompt the following sequence:

```
XPDL -- Please select dimension by its extension line:  
Please indicate where to break:
```

You are asked to select a linear or an angular dimension by one of its extension lines. It is the extension line nearest to the pick point be replaced by the real line entity. In fact, you may also select an existing line which is to break for the same escaping purpose.

After the extension line is replaced by a line, or an existing line is selected, the next prompt asks you to designate a point on the line to break. **XPDL** will issue the **INTER** snap directive automatically for you to help designating the break point, assuming you are to escape the line from other intersecting objects. You may press space bar to skip this breaking operation or issue other snap directive to snap a point other than intersection point.

If a break point is given, the line will be broken into two parts with a gap in between around it. The distance of this gap is determined by the system variable @DIMGAP.

**XPDL** will cycle through the same prompt sequence until you press the space bar to the first prompt (selecting nothing) or pressing <Ctrl/C> to terminate the operation.

**Special Notes:**

The **XPDL** command is an external command provided by the TCL program file "XPDL.TCL" or "XPDL.TCA". Either file must be present in the COMMANDS sub-directory where the

**TwinCAD** resides. If you can not issue **XPDL** command, you may solve the problem by copying the "XPDL.TCL" to the COMMANDS sub-directory.

**Procedure:**

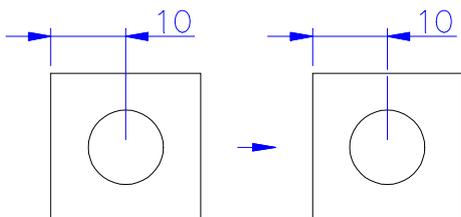
@CMD: XPDL (*return*)

XPDL -- Please select dimension by its extension line: (*do so*)

Please indicate where to break: INT of (*do so*)

...

**Example:**







## Create Y-Ordinate Dimension Command

---

### Purpose:

The **YDIM** command is used to create an Y-Ordinate dimension.

### Description:

The **YDIM** command lets you dimension designated points by their Y coordinates relative to a given origin. The dimension lines are always in horizontal direction.

When you issue the **YDIM** command, you will be asked to designate a point as the ordinate's relative origin for the subsequent dimensioning. After that, **TwinCAD** will continuously ask you to designate points to dimension and the locations to place the dimension texts. The dimension text for a designated point is automatically generated to reflect its true ordinate distance from the origin.

The placement of the dimension text can be automatic or by manual dragging, depending on current active option selected. The dimension line, starting from the designated point to the text, may be split into two parts which are joined by a vertical lead line. This allows a more flexible way in placing the dimension text and will avoid possible overlap between two adjacent dimensions.

The following lists the sub-command options:

- L**      **Lead**, specifying that an additional lead line is required. If this option is active, the dimension text can be dragged in both X and Y directions, and the dimension line will be split into two parts in the middle with a vertical lead line connecting both parts. After you have determined where to place the dimension text, **TwinCAD** will let you adjust the lead line location by dragging it horizontally.
- N**      **No Lead**, specifying that no lead lines will be generated for the dimension lines. If this option is active, the dimension text can only be dragged in X direction, and once its location is determined, the dimensioning is completed.
- D**      **Drag**, specifying that the text position is free to drag, not to be fixed.
- F**      **Fix**, specifying that the text position is fixed at the same X coordinate level as the last dimensioning. In the case when the Lead option is active, you can still drag the dimension text but only in the Y direction, as its X coordinate has been fixed. In normal case, however, the dimension text is placed automatically as soon as you designate the point to dimension. Note that you must explicitly determine the dimension text position of the first dimensioning.
- FL**     **Fix in Length**, specifying that the text position will be at the X coordinate level with a fixed horizontal distance from the designated point. That is, the length of the dimension line will be fixed and is the same as that for the last dimensioning. In the case when the Lead option is active, you can still drag the dimension text but only in the Y direction, as its X coordinate has been determined. Normally, however, the dimension text is placed automatically as soon as you designate the point to dimension.
- U**      **Undo**, request to remove the last dimensioning.
- C**      **Change-text**, request to change the dimension text.
- X**      **X-ordinate** dimensioning, request to switch to the X-ordinate dimensioning.

- Y**     **Y-ordinate** dimensioning, request to switch to the Y-ordinate dimensioning, meaningless as you are already in Y-ordinate dimensioning.
- A**     **Auto**, request to produce the ordinate dimensioning automatically.

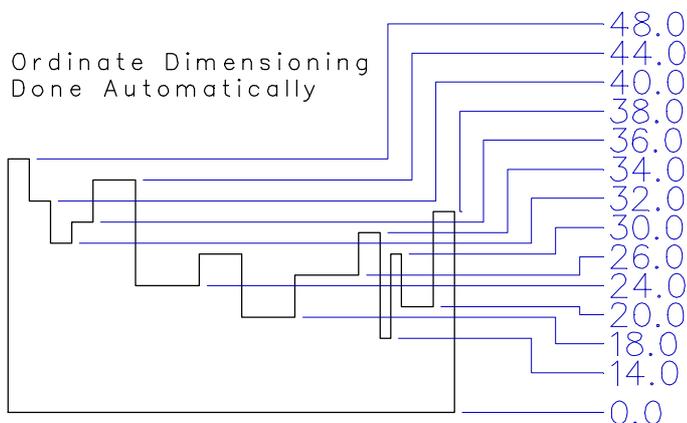
To exit the command, answer to the prompt with a null return.

See **XDIM** for **Automatic Ordinate Dimensioning**.

**Procedure:**

- **To generate the Y-ordinate dimension, follow the procedure below:**
  - @CMD: **YDIM** (return)
  - Indicate original point <0.,0.>: (point)
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: (point)
  - Lead/Nonlead/FLen/Fix/Drag/Undo/<text position>: (point)
  - To fix the dimensioning on the same level
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: **F** (return)
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: (point)
  - To undo the last dimensioning
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: **U** (return)
  - To dimension with a lead line
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: **L** (return)
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: (point)
  - Lead/Nonlead/FLen/Fix/Drag/Undo/<text position>: (point)
  - Settle lead lines location: (point)
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: (point)
  - ...
- **To generate dimension automatically**
  - Xdim/Ydim/Auto/Lead/Nonlead/FLen/Fix/Drag/Undo/(Y)<dimension point>: **A**(return)
  - Select Objects (+): (do so)
  - Generate dimension only on parallel segments ? <N>: (as you wish)
  - Please Indicate text position or length of lead line: (point)
  - nn generated. Use CHANGE command to adjust for possible conflicts.

**Example:**



**'ZOOM**

## Change Current View Window Command

---

**Purpose:**

The **ZOOM** command is used to change the current view window by its size and location.

**Description:**

The **ZOOM** command lets you change the size and location of the current view window over the drawing. You use this command to magnify the drawing to see the details, or to shrink it to view more or all of the drawing with less details.

The **ZOOM** command provides the following sub-command options:

- (point)** specifying the **new center location** of the view window. The current view window is moved to the new location.
  - (value)** specifying a **magnification value** relative to current view. A value less than 1 shrinks the drawing in the view window, while a value greater than 1 will magnify it. The view window changes as if you zoomed in or zoomed out the lens from a camera. The center of the view window is not changed.
  - A All**, specifying to set the current view window size and location such that all part of the drawing and the drawing limits are within the view window.
  - C Center**, request to specify the new center location and display height of the view window.
  - E Extents**, specifying to set the current view window size and location by the drawing extents, such that all part of the drawing can be viewed on the display as large as possible.
- Note: You may use **ZESCALE** system variable to specify desired scale factor of ZE view window. For example, set ZESCALE = 0.9 will reduce the view window by a factor of 0.9.
- L Left**, request to set the current view window by specifying the lower left corner and new display height.
  - P Previous**, request to restore previous view window.
  - N Next**, request to reverse the last **ZOOM/Previous** operation.
  - W Window**, request to set the current view window size and location by designating a rectangular window area (two points).

After the new view window has been set up, **TwinCAD** will redraw the drawing on the new view window.

**Procedure:**

- **To magnify a portion of the drawing in the window:**
  - @CMD: **ZOOM** (*return*)
  - All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: **W** (*return*)
  - First corner: (*point*)
  - Second corner: (*point*)
- **To pan the view window by new center position:**

@CMD: **ZOOM** (return)

All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: (point)

- **To Zoom/IN the view by a magnification factor of 2:**

@CMD: **ZOOM** (return)

All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: **2.** (return)

- **To view all the drawing by its extents:**

@CMD: **ZOOM** (return)

All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: **E** (return)

- **To view back the previous view:**

@CMD: **ZOOM** (return)

All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: **P** (return)

- **To set new view window location and size:**

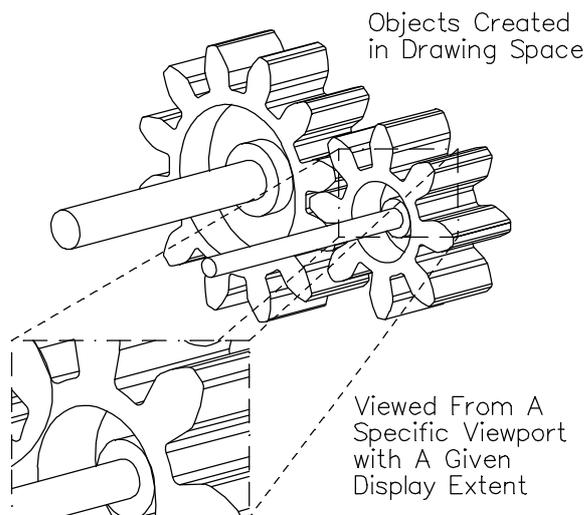
@CMD: **ZOOM** (return)

All/Center/Extents/Left/Previous/Window/Next/<pan center/Scale>: **C** (return)

Center point: (point)

Display Height: (value)

### Example:



## Short Name for Zooming

---

### Purpose:

The **Zx** {I,O,W,E,P,N} command performs a zoom operation.

### Description:

These commands are short names for the zoom operations described below: (See **ZOOM** command for details.)

**ZI** Zoom/In, equivalent to **ZOOMIN**

**ZO** Zoom/Out, equivalent to **ZOOMOUT**

**ZW** Zoom/Window, equivalent to **ZOOMW**

**ZE** Zoom/Extent, equivalent to **ZOOMEXT**

**ZP** Zoom/Previous, equivalent to **ZOOMPREV**

**ZN** Zoom/Next, equivalent to **ZOOMNEXT**

### Example:

