Tecplot®

Reference Manual

Version 10 Release 4

Tecplot, Inc.
Bellevue, Washington
November, 2004

SOFTWARE COPYRIGHTS

TRADEMARKS

NOTICE TO U.S. GOVERNMENT END-USERS

CONTENTS

PART I

Macro Command Language

CHAPTER 1 Introduction

A Tecplot macro is a set of instructions, called macro commands, which perform actions in Tecplot. Macro commands can be used to accomplish virtually any task that can be done via the Tecplot interface, offering an easy way to automate Tecplot processes. The only things you can do interactively that cannot be done with macro commands are those actions that have no effect on a final, printed plot (such as resizing the Tecplot process window). To augment this ability, there are macro commands which have no corresponding interactive control, such as looping and conditional commands. These commands typically go hand in hand with the execution of a macro.

You can create macros by recording them from the Tecplot interface using the Macro Recorder, or create them from scratch using any ASCII text editor. In most cases, the most effective approach to creating a macro is the following hybrid approach:

1. Run Tecplot and choose to record a macro to a file. Perform tasks similar to those you are trying to capture in the final macro.
2. Close the recording session and examine the macro file. The commands generated by Tecplot should be fairly readable and easy to understand.
3. Make minor modifications to the recorded macro. Typical modifications involve adding loops, adding variables, or adding commands that, for example, prompt the user to enter a file name.

One of the main reasons for using the approach above is the large number of commands and permutations of parameters. This manual provides an exhaustive listing of the available macro commands. However, it is often easier to have Tecplot perform the action and record the relevant command than look up individual commands and their required parameters.

An important feature of Tecplot's macro command language is its Viewer/Debugger. Often, you will have a well-developed macro that needs some modification. You can use the Debugger to step through the macro to the point where you want the change to be made and then start recording to a new file. Using a text editor, you can insert macro commands from a new file into an existing macro file.

CHAPTER 2 ***Managing Macros***

Tecplot macros are stored in files. These files are processed by loading them into Tecplot and running them.

2.1. Macros vs. Macro Functions vs. Macro Commands

A Tecplot macro is a file containing one or more macro commands. These files start with the following special comment line to notify Tecplot that what follows is a Tecplot Version 10 macro:

```
#!MC 1000
```

Any number of macro commands or comments may follow.

Tecplot macro functions are defined are defined in Tecplot macros by using the **`$!MACROFUNCTION-$!ENDMACROFUNCTION`** commands. Between the **`$!MACROFUNCTION and $!ENDMACROFUNCTION`** commands you may use any valid macro command (except **`$!MACROFUNCTION`**). When a Tecplot macro is loaded, all macro functions are extracted and the attached commands are not executed until a **`$!RUNMACROFUNCTION`** command is encountered.

Macro functions may be retained if desired. A retained macro function remains defined in Tecplot even if the macro in which it was defined is replaced by another macro. Retained macro functions may be called by other macros that are loaded at a later time.

2.2. Running Macros from the Command Line

A simple way to run a Tecplot macro is to include it in the command line with the **`-p`** flag. The following command runs Tecplot and plays a macro called **`a.mcr`**:

```
tecplot -p a.mcr
```

If you use the **`.mcr`** extension for the macro file name, then the **`-p`** flag is optional. If you want to debug the macro, include the **`-z`** flag as well.

2.3. Running Macros from the Tecplot Interface

You can run a macro file by going to the File menu and selecting the Macro sub-menu, followed by the Play option. A dialog appears; choose the macro to play.

If you want to debug a macro file, go to the File menu and selecting the Macro sub-menu, followed by the View option. The Macro Viewer dialog appears so you can load in a macro. When the macro is loaded, Tecplot waits at the first macro command for you to step through the commands. See the *Tecplot User's Manual* for complete details on how to use the Macro Viewer.

2.4. Running Macros from the Quick Macro Panel

Macros that you use frequently or want rapid access to may be defined as macro functions within a special file called **tecplot.mcr** in either the current directory, your home directory, or the Tecplot home directory. When Tecplot starts it looks for this file in each of those directories in turn. If Tecplot finds the file, it loads the macro definitions and associates functions to buttons on the Quick Macro Panel (in the Tools menu). You can have Tecplot load your own macro function file by using the **-qm** flag on the command line. The following command runs Tecplot and installs the macro functions in the file **myteccmd.mcr** into the Quick Macro Panel:

```
    tecplot -qm myteccmd.mcr
```

You can have a macro function add a button to the Quick Macro Panel. By default, all macro functions defined in the **tecplot.mcr** file will add a button to the Quick Macro Panel, those defined elsewhere will not. See the **$!MACROFUNCTION** command for more information.

If you want Tecplot to display the Quick Macro Panel at starting include the **-showpanel** flag on the command line.

To see an example of a macro function file, look at the file **tecplot.mcr** located in the **examples/mcr** sub-directory below the Tecplot home directory. If this file is moved to the Tecplot home directory, the Quick Macro Panel will have options that include 3D Rotation Animation and Reset Center of Rotation.

CHAPTER 3 Macro Command Syntax

A macro file consists of one or more macro commands. Comments may be inserted anywhere in the file, except within a character string. Comments start with an "**#**" (octothorp) and extend to the end of the line. The first line of a macro file contains a special comment that identifies the version number of the macro file. For Tecplot Version 10, this line is **#!MC 1000.**

A Tecplot Version 10 macro file has the form:

```
#!MC 1000
 <macrocommand>
 <macrocommand>

            .  .  .
```

Each *macrocommand*, in turn, has the form:

```
$!commandname [commandspecificmodifiers]
[mandatoryparameters]
[optionalparameters]
```

where

commandspecificmodifiers These are optional command-specific modifiers. An example of a command that uses this is the $!FIELD command. The $!FIELD command can be followed by a "set." If it is not followed by a set, the $!FIELD command applies to all enabled zones. A supplied set in this case is used to limit the zones to which the $!FIELD command applies.

mandatoryparameters *commandparameter commandparameter...*

optionalparameters *commandparameter commandparameter...*

commandparameter *parameterassignment or parametersubcommand.*

parameterassignment *parametername op value.*

op = or -= or += or *= or /=.

parametersubcommand *parametername {optionalparameters}.*

commandname The name of a major command, such as **REDRAW**.

parametername The name of a valid parameter for the previously named major command. For example, the $!REDRAW major command has an optional parameter called DOFULLDRAWING.

value — *number, expression,* or `enumeratedvalue`.

number — Any valid integer or double value representation.

expression — Any valid infix notation expression. The entire expression must itself be enclosed in parenthesis. For example (3+5).

enumeratedvalue — A key word that is unique to the variable being assigned a value. For example, if the variable being assigned a value is a basic color then the enumerated value can be one of the following: **BLACK, RED, GREEN, BLUE, CYAN, YELLOW, PURPLE, WHITE, CUSTOM1** through CUSTOM56.

Spacing and capitalization for macro commands are, for the most part, not important. The following examples show different ways to enter the same macro command to set the width and height for the custom1 paper:

Example 1:

```
$!PAPER
   PAPERSIZEINFO
   {
    CUSTOM1
    {
     WIDTH = 3
    }
   }
```

Example 2:

```
$!PAPER PAPERSIZEINFO
   {CUSTOM1
    {WIDTH = 3}
   }
```

Example 3:

```
$!paper papersizeinfo {custom1 {width = 3}}
```

CHAPTER 4 *Macro Command Summary*

This chapter presents a brief list of the major macro commands in Tecplot. All major macro commands are preceded by "**$!**" (dollar sign, exclamation mark).

The macro commands fall into three basic categories:

- Control commands (Control in the Type column) deal with the flow of control within a Tecplot macro.
- Action commands (Action in the Type column) perform some type of visible action in Tecplot like rotating an object or redrawing a frame, file input/output, or creating or destroying objects within Tecplot.
- SetValue commands (FSV in the Type column refers to Frame SetValue commands; GSV to General SetValue) assign values to change the state of Tecplot. Some values change the state of the current frame; others are more general and are used to change the settings of the interface or hardcopy output from Tecplot. SetValue commands are hierarchical in nature.

Command	Description	Type
$!ACTIVEFIELDZONES	Change the set of active zones.	**FSV**
$!ACTIVELINEMAPS	Change the set of active Line-maps.	**FSV**
$!ADDMACROPANELTITLE	Add a title to the Quick Macro Panel.	**Action**
$!ADDONCOMMAND	Execute command in an **add-on.**	**Action**
$!ALTERDATA	Execute an equation to alter data.	**Action**
$!ANIMATECONTOURLEVELS	Show an animation of contour levels.	**Action**
$!ANIMATEIJKBLANKING	Show an animation of IJK-blanking.	**Action**
$!ANIMATEIJKPLANES	Show an animation of IJK-planes.	**Action**
$!ANIMATESLICES	Show an animation of currently defined slices.	**Action**
$!ANIMATESTREAM	Show an animation of stream time marks or dashes.	**Action**
$!ANIMATELINEMAPS	Show an animation of Line-mappings.	**Action**
$!ANIMATEZONES	Show an animation of zones.	**Action**
$!ATTACHDATASET	Attach a data set to the current frame.	**Action**
$!ATTACHGEOM	Attach a geometry to the current frame.	**Action**
$!ATTACHTEXT	Attach a text to the current frame.	**Action**

Command	Description	Type
$!AVERAGECELLCENTERDATA	Interpolate cell-centered data to cell nodes.	Action
$!BASICCOLOR	Change the RGB values for basic colors.	GSV
$!BASICSIZE	Change drop-down menu size defaults for things like fonts, symbols, line thicknesses, and so forth.	GSV
$!BLANKING	Change value or IJK-blanking settings.	FSV
$!BRANCHCONNECTIVITY	Branch connectivity data from a zone.	FSV
$!BRANCHFIELDDATAVAR	Branch a variable from sharing in a zone.	FSV
$!BREAK	Break out of current $!LOOP or $!WHILE.	Control
$!COLORMAP	Change the color map settings.	GSV
$!COLORMAPCONTROL	Perform operations on the color map.	Action
$!COMPATIBILITY	Backward compatibility settings.	GSV
$!CONTINUE	Continue to end of current $!LOOP or $!WHILE.	Control
$!CONTOURLABELS	Add or delete contour labels.	Action
$!CONTOURLEVELS	Add, delete, or reset the contour levels.	Action
$!CREATECIRCULARZONE	Create a circular or cylindrical zone (2- or 3-D).	Action
$!CREATECONTOURLINEZONES	Create a zone or zones from contour lines.	Action
$!CREATEFEBOUNDARY	Create an FE-boundary zone.	Action
$!CREATEFESURFACEFROMIORDERED	Create an FE-surface from two or more I-Ordered zones.	Action
$!CREATEISOZONES	Create iso-surface zones.	Action
$!CREATELINEMAP	Create a Line-mapping.	Action
$!CREATEMIRRORZONES	Create mirror-image zones.	Action
$!CREATENEWFRAME	Create a new frame.	Action
$!CREATERECTANGULARZONE	Create a rectangular or cubical zone (2- or 3-D).	Action
$!CREATESIMPLEZONE	Create a simple zone.	Action
$!CREATESLICEZONEFROMPLANE	Create a zone by slicing a volume zone.	Action
$!CREATESLICEZONES	Create a new zone for each slice defined on the Slice Details dialog.	Action
$!CREATESTREAMZONES	Create streamtrace zones.	Action
$!DATASETUP	Miscellaneous scratch data and Preplot setup.	GSV
$!DEFAULTGEOM	Change the default geometry settings.	GSV
$!DEFAULTTEXT	Change the default text settings.	GSV

Command	Description	Type
$!DELAY	Delay execution of Tecplot.	**Action**
$!DELETEAUXDATA	Delete auxiliary data attached to specified object.	**Action**
$!DELETELINEMAPS	Delete Line-mappings.	**Action**
$!DELETEVARS	Delete variables.	**Action**
$!DELETEZONES	Delete zones.	**Action**
$!DOUBLEBUFFER	Enable or disable double buffering or swap buffers.	**Action**
$!DRAWGRAPHICS	Enable or disable drawing of graphics to the screen.	**Action**
$!DROPDIALOG	Drop a dialog (see $!LAUNCHDIALOG).	**Action**
$!DUPLICATELINEMAP	Duplicate an Line-mapping.	**Action**
$!DUPLICATEZONE	Duplicate a zone.	**Action**
$!ELSE	Conditionally handle macro commands.	**Action**
$!ELSEIF	Conditionally handle macro commands.	**Action**
$!ENDIF	End of $!IF-$!ENDIF construct.	**Control**
$!ENDLOOP	End of $!LOOP-$!ENDLOOP construct.	**Control**
$!ENDMACROFUNCTION	End of $!MACROFUNCTION-$!ENDMACROFUNCTION construct.	**Control**
$!ENDWHILE	End of $!WHILE-$!ENDWHILE construct.	**Control**
$!EXPORT	Export the current plot to a file.	**Action**
$!EXPORTCANCEL	Cancel the current export.	**Action**
$!EXPORTFINISH	Signals completion of an animation sequence.	**Action**
$!EXPORTNEXTFRAME	Records the next frame of an animation.	**Action**
$!EXPORTSETUP	Change the file export settings.	**GSV**
$!EXPORTSTART	Signals the start of an animation sequence.	**Action**
$!EXTRACTFROMGEOM	Extract data from points along a polyline geometry.	**Action**
$!EXTRACTFROMPOLYLINE	Extract data from a supplied polyline.	**Action**
$!FIELD	Change the plot style settings for zones.	**FSV**
$!FIELDLAYERS	Change the active layers for field plots.	**FSV**
$!FILECONFIG	Change miscellaneous file path configuration settings.	**GSV**
$!FONTADJUST	Change intercharacter spacing, subscript, and superscript sizing, and so forth.	**GSV**
$!FRAMECONTROL	Push, pop, or delete frames.	**Action**

Command	Description	Type
$!FRAMELAYOUT	Change size, position, and so forth of the current frame.	FSV
$!FRAMENAME	Change the frame name.	FSV
$!FRAMESETUP	Change miscellaneous default frame style settings.	GSV
$!GETAUXDATA	Retrieve auxiliary data from an object.	Action
$!GETCONNECTIVITYREFCOUNT	Get the number of zone shared with a zone.	Action
$!GETCURFRAMENAME	Get the name of the current frame.	Action
$!GETFIELDVALUE	Get the field value at a specified point index, and assign it to <macrovar>.	Action
$!GETFIELDVALUEREFCOUNT	Get the count of how many places a variable is shared.	Action
$!GETNODEINDEX	Get the specified node index for finite-element zones.	Action
$!GETVARLOCATION	Returns the variable location. Node or Cell-Centered.	Action
$!GETVARNUMBYNAME	Get the position of a variable.	Action
$!GETZONETYPE	Get the zone type of specified zone.	Action
$!GLOBALCONTOUR	Change global contour settings.	FSV
$!GLOBALFRAME	Change miscellaneous global frame settings.	GSV
$!GLOBALISOSURFACE	Change global attributes associated with isosurfaces.	FSV
$!GLOBALLINEPLOT	Change global Line-plot settings.	FSV
$!GLOBALPOLAR	Change global settings of polar plots	FSV
$!GLOBALRGB	Change Global RGB coloring	FSV
$!GLOBALSCATTER	Change global scatter settings.	FSV
$!GLOBALSLICE	Change global attributes associated with slices.	FSV
$!GLOBALSTREAM	Change global streamtrace settings.	FSV
$!GLOBALTHREED	Change global 3-D settings.	FSV
$!GLOBALTHREEDVECTOR	Change global 3-D vector settings.	FSV
$!GLOBALTWODVECTOR	Change global 2-D vector settings.	FSV
$!IF	Conditionally execute macro commands.	Control
$!INCLUDEMACRO	Include macro commands from another file.	Control
$!INTERFACE	Change interface settings.	GSV
$!INVERSEDISTINTERPOLATE	Interpolate data using the inverse distance algorithm.	Action

Command	Description	Type
`$!KRIG`	Interpolate data using kriging.	`Action`
`$!LAUNCHDIALOG`	Launch a dialog (see $!DROPDIALOG).	`Action`
`$!LIMITS`	Change limits for lines, text length, and contour levels.	`GSV`
`$!LINEARINTERPOLATE`	Interpolate data using linear interpolation.	`Action`
`$!LINEMAP`	Change plot style settings for Line-maps.	`FSV`
`$!LINEPLOTLAYERS`	Turn Line-plot layers and features on or off.	`FSV`
`$!LINKING`	Link attributes in two or more frames so that changes to attributes of one frame effect all linked frames.	`FSV`
`$!LOADADDON`	Load an add-on.	`Action`
`$!LOADCOLORMAP`	Load a color map from a file.	`Action`
`$!LOOP`	Begin a loop in a macro.	`Control`
`$!MACROFUNCTION`	Begin definition of a macro function.	`Control`
`$!NEWLAYOUT`	Clear the current layout and start over.	`Action`
`$!OPENLAYOUT`	Open and read in a layout file.	`Action`
`$!PAPER`	Change paper settings.	`GSV`
`$!PAUSE`	Pause the macro and display a message.	`Action`
`$!PICK`	Select and operate on objects.	`Action`
`$!PLOTTYPE`	Change between view modes.	`FSV`
`$!POLARAXIS`	Control axis settings for polar plots.	`FSV`
`$!POLARTORECTANGULAR`	Convert coordinate variables from polar to rectangular.	`Action`
`$!POLARVIEW`	Set the extents of polar plots.	`GSV`
`$!PRINT`	Print the current layout to the system spooler or to a file.	`Action`
`$!PRINTSETUP`	Change printing settings.	`GSV`
`$!PROMPTFORFILENAME`	Launch a file selection dialog.	`Action`
`$!PROMPTFORTEXTSTRING`	Launch a dialog containing a text string and optional instructions.	`Action`
`$!PROMPTFORYESNO`	Launch a dialog containing “yes” and “no” buttons.	`Action`
`$!PROPAGATELINKING`	Link multiple frames.	`FSV`
`$!PUBLISH`	Create an HTML file displaying one or more images. A linked layout with packaged data may be included.	`Action`

Command	Description	Type
$!QUIT	Quit Tecplot.	Action
$!RAWCOLORMAP	Install a raw color map.	Action
$!READDATASET	Load a data set by reading in one or more data files.	Action
$!READSTYLESHEET	Read a stylesheet into the current frame.	Action
$!REDRAW	Redraw the current frame.	Action
$!REDRAWALL	Redraw all frames.	Action
$!REMOVEVAR	Remove a user-defined macro variable.	Control
$!RENAMEDATASETVAR	Rename a data set variable.	Action
$!RENAMEDATASETZONE	Rename a data set zone.	Action
$!RESET3DAXES	Reset the 3-D axes.	Action
$!RESET3DORIGIN	Reset the 3-D origin to the centroid of the data.	Action
$!RESET3DSCALEFACTORS	Reset the 3-D axes' scale factors	Action
$!RESETVECTORLENGTH	Reset the vector length.	Action
$!ROTATE2DDATA	Rotate 2-D data. This alters the data set.	Action
$!ROTATE3DVIEW	Rotate a 3-D object.	Action
$!RUNMACROFUNCTION	Run a macro function.	Control
$!SAVELAYOUT	Save the layout to a file.	Action
$!SET3DEYEDISTANCE	Set view distance from the current center of rotation.	FSV
$!SETAUXDATA	Add auxiliary data to an object.	GSV
$!SETDATASETTITLE	Set the data set title.	Action
$!SETFIELDVALUE	Change the value of a field variable for a specific index and zone.	Action
$!SETSTYLEBASE	Set which attributes are used to build new frames.	Action
$!SHARECONNECTIVITY	Share nodemaps between zones	GSV
$!SHAREFIELDDATAVAR	Share variables between zones	GSV
$!SHIFTLINEMAPSTOBOTTOM	Shift Line-mappings to the bottom (making them draw later).	Action
$!SHIFTLINEMAPSTOTOP	Shift Line-mappings to the top (making them draw earlier).	Action
$!SHOWMOUSEPOINTER	Activate mouse icon within a macro.	Action
$!SKETCHAXIS	Change sketch axis settings.	FSV
$!SMOOTH	Smooth data.	Action

Command	Description	Type
$!STREAMTRACE	Add or delete streamtraces. Define the termination line.	**Action**
$!SYSTEM	Execute an operating system command.	**Action**
$!THREEDAXIS	Change 3-D axis settings.	**FSV**
$!THREEDVIEW	A SetValue command that changes global attributes associated with the 3-D view.	**FSV**
$!TRANSFORMCOORDINATES	Transform coordinates from one plot style to another.	**FSV**
$!TRIANGULATE	Create a new zone by triangulating data from existing zones.	**Action**
$!TWODAXIS	Change 2-D axis settings.	**FSV**
$!VARSET	Assign a value to a user-defined macro variable.	**Control**
$!VIEW	Change the view in the current frame.	**Action**
$!WHILE	Begin a WHILE loop in a macro.	**Control**
$!WORKSPACEVIEW	Change the view of the frames in the workspace.	**Action**
$!WRITECOLORMAP	Write the current color map to a file.	**Action**
$!WRITECURVEINFO	Write coefficients or data points for curve fits in XY-plots to a file.	**Action**
$!WRITEDATASET	Write the data set for the current frame to a file.	**Action**
$!WRITESTYLESHEET	Write a stylesheet for the current frame to a file.	**Action**
$!XYLINEAXIS	Change XY-plot axis settings.	**FSV**

CHAPTER 5 ***Macro Commands***

This chapter lists Tecplot's macro commands alphabetically. Items within double angle brackets (**<< >>**) represent parameter sub-commands that are listed and described in Chapter 6.

$!ACTIVEFIELDZONES

Syntax: **`$!ACTIVEFIELDZONES`** *<op> <set>*
[no parameters]

Description: A SetValue command that changes the set of zones considered for plotting.

Examples:

Example 1: Make only zones 1, 3, 4 and 5 active for plotting:

```
$!ACTIVEFIELDZONES = [1,3-5]
```

Example 2: Add zones 33, 34, 35 and 36 to the set of active zones:

```
$!ACTIVEFIELDZONES + = [33-36]
```

Example 3: Remove zones 1, 2, 3, 9, 10 and 11 from the set of active zones:

```
$!ACTIVEFIELDZONES - = [1-3,9-11]
```

$!ACTIVELINEMAPS

Syntax: **`$!ACTIVELINEMAPS`** *<op> <set>*
[no parameters]

Description: A SetValue command that changes the set of line-mappings considered for plotting.

Examples:

Example 1: Make only line-mappings 1, 3, 4 and 5 active for plotting:

```
$!ACTIVELINEMAPS = [1,3-5]
```

Example 2: Add line-maps 33, 34, 35 and 36 to the set of active line-mappings:

```
$!ACTIVELINEMAPS + = [33-36]
```

Example 3: Remove line-maps 1, 2, 3, 9, 10 and 11 from the set of active line-mappings:

```
$!ACTIVELINEMAPS - = [1-3,9-11]
```

$!ADDMACROPANELTITLE

Syntax: **$!ADDMACROPANELTITLE** <*string*>
[no parameters]

Description: Add a title to the Quick Macro Panel.

Example: The following example adds the title “Bar Charts” to the Quick Macro Panel:

```
$!ADDMACROPANELTITLE "Bar Charts"
```

$!ADDONCOMMAND

Syntax:

```
$!ADDONCOMMAND
     ADDONID = <string>
     COMMAND = <string>
      [optional parameters]
```

Description: Send a command to an add-on. An add-on registers the name of a function that will be called when an **$!ADDONCOMMAND** is processed. Tecplot knows which registered function to call based on the **ADDONID** string. See the function **TecUtilMacroAddCommandCallback** in the *Tecplot ADK Reference Manual*.

Required Parameters:

Parameter Syntax	Notes
ADDONID = <string>	String that identifies the add-on. This must match the published ID string for the add-on.
COMMAND = *<string>*	The command to be sent to the add-on.

Optional Parameters:

Parameter Syntax	Default	Notes
<addoncommandrawdata>	**NULL**	If the **RAWDATA** section is supplied then each line of the **RAWDATA** section is appended to the **COMMAND** string. A leading new line character is appended first, and each line in the **RAWDATA** section will also be terminated with a new line (except for the last line).

Example: Send the command **GO** to the add-on that has registered a command processor with an add-on ID of **XPROC**:

```
$!ADDONCOMMAND
   ADDONID = "XPROC"
   COMMAND = "GO"
```

$!ALTERDATA

Syntax:

```
$!ALTERDATA <set>
   EQUATION = <string>
   [optional parameters]
```

Description: The **ALTERDATA** function operates on a data set within Tecplot using FORTRAN-like equations. See the *Tecplot User's Manual* for more information on using equations in Tecplot. The *<set>* parameter, if specified, represents the set of zones on which to operate. If *<set>* is omitted, all zones are affected.

Required Parameter:

Parameter Syntax	Notes
EQUATION = *<string>*	This assigns the equation to use to operate on the data.

Optional Parameters:

Parameter Syntax	Default	Notes

`IRANGE`
`{`
` MIN  = <integer>`
` MAX  = <integer>`
` SKIP = <integer>`
`}`
	`1`
	`0`
	`1`
		See the note, Range Parameters, for information on specifying range index values.

`JRANGE`
`{`
` MIN  = <integer>`
` MAX  = <integer>`
` SKIP = <integer>`
`}`
	`1`
	`0`
	`1`
		See the note, Range Parameters, for information on specifying range index values.

`KRANGE`
`{`
` MIN  = <integer>`
` MAX  = <integer>`
` SKIP = <integer>`
`}`
	`1`
	`0`
	`1`
		See the note, Range Parameters, for information on specifying range index values.

`DATATYPE = <datatype>`	`SINGLE`	Assign the precision given to the destination variable (that is, the variable on the left hand side of the equation). This only applies if the equation creates a new variable. (see Example 2).
`VALUELOCATION =` *<valuelocation>*	`AUTO`	Assign the location to destination variable.

Range Parameters

The **IRANGE, JRANGE**, and **KRANGE** parameters are used to limit the data altered by the equation. The specification of range indices follow these rules:

- All indices start with 1 and go to some maximum index *m*.
- The number **0** can be used to represent the maximum index *m*; specifying **0** tells the command to go to the very last position of the range, that is, the maximum index value *m*. If the maximum index *m* = 15, specifying **0** sets the range index to 15.
- Negative values represent the offset from the maximum index. If a value of -2 is specified, and the maximum index *m* is 14, the value used is 14-2, or 12.

Examples:

Example 1: The following example adds one to X for all zones for every data point:

```
$!ALTERDATA
   EQUATION = "x = x+1"
```

Example 2: The following example creates a new, double precision variable called **DIST**:

```
$!ALTERDATA
   EQUATION = "{DIST} = SQRT(X**2 + Y**2)"
   DATATYPE = DOUBLE
```

Example 3: The following equations set a variable called **P** to zero along the boundary of an IJ-ordered zone:

```
$!ALTERDATA
   EQUATION = "{P} = 0"
   IRANGE {MAX = 1}
$!ALTERDATA
   EQUATION = "{P} = 0"
   IRANGE {MIN = 0}
$!ALTERDATA
   EQUATION = "{P} = 0"
   JRANGE {MAX = 1}
$!ALTERDATA
   EQUATION = "{P} = 0"
   JRANGE {MIN = 0}
```

$!ANIMATECONTOURLEVELS

Syntax:

```
$!ANIMATECONTOURLEVELS
    START = <integer>
    END   = <integer>
  [optional parameters]
```

Description: Produce an animation of a contour line plot by showing a single level at a time. The animation varies according to the currently defined contour levels and is limited by the values in the **START**, **END**, and **SKIP** parameters. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameters:

Parameter Syntax	Notes
START = *<integer>*	Starting contour level number to animate.
END = *<integer>*	Ending contour level number to animate.

Optional Parameters:

Parameter Syntax	Default	Notes
`SKIP = <integer>`	1	Level skip.
`CREATEMOVIEFILE = <boolean>`	FALSE	If **TRUE**, must be preceded by **$!EXPORTSETUP** commands.

Example: The following command animates the first four contour levels to an AVI file:

```
$!EXPORTSETUP EXPORTFORMAT = AVI
$!EXPORTSETUP EXPORTFNAME = "contourlevels.avi"
$!ANIMATECONTOURLEVELS
  START = 1
  END   = 4
  CREATEMOVIEFILE = TRUE
```

$!ANIMATEIJKBLANKING

Syntax:

```
$!ANIMATEIJKBLANKING
    NUMSTEPS = <integer>
    [optional parameters]
```

Description: Produce an animation of different IJK-blankings in your plot. The animation starts at one IJK-blanking setting and marches through intermediate steps to a second setting. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameter:

Parameter Syntax	Notes
`NUMSTEPS = <integer>`	Number of intermediate steps for the animation.

Optional Parameters:

Parameter Syntax	Default	Notes
`IMINFRACT = <dexp>`	0.1	Minimum fraction for blanking at the start of animation for the I-index. Actual I-index is equal to **IMINFRACT*IMAX**.
`JMINFRACT = <dexp>`	0.1	Minimum fraction for blanking at the start of animation for the J-index. Actual J-index is equal to **JMINFRACT*JMAX**.

Parameter Syntax	Default	Notes
KMINFRACT = *<dexp>*	**0.1**	Minimum fraction for blanking at the start of animation for the K-index. Actual K-index is equal to **KMINFRACT*KMAX**.
IMAXFRACT = *<dexp>*	**1.0**	Maximum fraction for blanking at the start of animation for the I-index. Actual I-index is equal to **IMAXFRACT*IMAX**.
JMAXFRACT = *<dexp>*	**1.0**	Maximum fraction for blanking at the start of animation for the J-index. Actual J-index is equal to **JMAXFRACT*JMAX**.
KMAXFRACT = *<dexp>*	**1.0**	Maximum fraction for blanking at the start of animation for the K-index. Actual K-index is equal to**KMAXFRACT*KMAX**.
IMINFRACT2 = *<dexp>*	**0.8**	Minimum fraction for blanking at the end of animation for the I-index. Actual I-index is equal to **IMINFRACT*IMAX**.
JMINFRACT2 = *<dexp>*	**0.8**	Minimum fraction for blanking at the end of animation for the J-index. Actual J-index is equal to **JMINFRACT*JMAX**.
KMINFRACT2 = *<dexp>*	**0.8**	Minimum fraction for blanking at the end of animation for the K-index. Actual K-index is equal to **KMINFRACT*KMAX**.
IMAXFRACT2 = *<dexp>*	**1.0**	Maximum fraction for blanking at the end of animation for the I-index. Actual I-index is equal to **IMAXFRACT*IMAX**.
JMAXFRACT2 = *<dexp>*	**1.0**	Maximum fraction for blanking at the end of animation for the J-index. Actual J-index is equal to **JMAXFRACT*JMAX**.
KMAXFRACT2 = *<dexp>*	**1.0**	Maximum fraction for blanking at the end of animation for the K-index. Actual K-index is equal to **KMAXFRACT*KMAX**.
CREATEMOVIEFILE = *<boolean>*	**FALSE**	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example produces an animation showing a band of I-planes traversing the entire data field:

```
$!ANIMATEIJKBLANKING
   NUMSTEPS       = 6
   IMINFRACT      = 0.1
   JMINFRACT      = 0.0
   KMINFRACT      = 0.0
   IMAXFRACT      = 1.0
   JMAXFRACT      = 1.0
   KMAXFRACT      = 1.0
   IMINFRACT2     = 1.0
   JMINFRACT2     = 0.0
```

```
KMINFRACT2    = 0.0
IMAXFRACT2    = 1.0
JMAXFRACT2    = 1.0
KMAXFRACT2    = 1.0
```

$!ANIMATEIJKPLANES

Syntax:

```
$!ANIMATEIJKPLANES
   START = <integer>
     END   = <integer>
  [optional parameters]
```

Description: Produce an animation that cycles through I-, J- or K-planes in an IJK-ordered data set. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameters:

Parameter Syntax	Notes
START = *<integer>*	Starting plane index.
END = *<integer>*	Ending plane index.

Optional Parameters:

Parameter Syntax	Default	Notes
PLANES = *<ijkplane>*	**I**	Specify I, J or K.
SKIP = *<integer>*	**1**	Index skip.
CREATEMOVIEFILE = *<boolean>*	**FALSE**	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example generates an animation of the I-planes 1, 3, 5, 7 and 9:

```
$!ANIMATEIJKPLANES
  PLANES = I
  START  = 1
  END    = 9
  SKIP   = 2
```

$!ANIMATELINEMAPS

Syntax:

```
$!ANIMATELINEMAPS
  START = <integer>
  END = <integer>
   [optional parameters]
```

Description: Produce an animation of one Line-mapping at a time. To create an AVI or RM file, add `$!EXPORTSETUP` commands before this command.

Required Parameters:

Parameter Syntax	Notes
`START = <integer>`	Starting Line-map number.
`END = <integer>`	Ending Line-map number.

Optional Parameters:

Parameter Syntax	Default	Notes
`SKIP = <integer>`	1	Line-map skip.
`CREATEMOVIEFILE = <boolean>`	`FALSE`	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example creates an animation showing plots of Line-maps 2, 4, 6, 8 and 10:

```
$!ANIMATELINEMAPS
   START = 2
   END   = 10
   SKIP  = 2
```

$!ANIMATESLICES

Syntax:

```
$!ANIMATESLICES
    START = <integer>
    END = <integer>
  [optional parameters]
```

Description: The macro command **$!ANIMATESLICES** uses the currently defined start and end slice position. Use **$!GLOBALSLICE** to set these positions; **$!ANIMATESLICES** then redefines how many intermediate slices are to be used, then animates a subset of those slices. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameters:

Parameter Syntax	Default	Notes
START = *<integer>*		Start and end indices are based on the set of slices generated by NUMSLICES. All slices between start and end are animated. There is no skipping. To obtain the effect of skipping, change the value for NUMSLICES.
END = *<integer>*		Start and end indices are based on the set of slices generated by NUMSLICES. All slices between start and end are animated. There is no skipping. To obtain the effect of skipping, change the value for NUMSLICES.
NUMSLICES = *<integer>*	**2**	Number of slices to distribute between the start and end slice locations as defined by POSITION1 and **POSITION2** in **$!GLOBALSLICE**.

Optional Parameters:

Parameter Syntax	Default	Notes
CREATEMOVIEFILE = *<boolean>*	**FALSE**	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example creates an animation of 3-D slices:

```
$!ANIMATESLICES
   START = 1
   END = 30
   NUMSLICES = 30
```

$!ANIMATESTREAM

Syntax:

```
$!ANIMATESTREAM
   [optional parameters]
```

Description: Produce an animation of stream markers or dashes, moving along the currently defined streamtrace paths. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Optional Parameters:

Parameter Syntax	Default	Notes
STEPSPERCYCLE = *<integer>*	10	Number of steps to use for each cycle of the animation. Increase this number to produce a smoother animation.
NUMCYCLES = *<integer>*	4	Number of cycles in the animation. Each cycle shows stream markers or dashes, moving along a streamtrace path. If DT is the streamtrace delta time, then at the end of the cycle, the markers or dashes will have moved **(2*DT*(STEPSPERCYCLE-1))/(STEPSPERCYCLE)** in time.
CREATEMOVIEFILE = *<boolean>*	**FALSE**	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example animates streamtraces for five cycles with each cycle using ten steps:

```
$!ANIMATESTREAM
   STEPSPERCYCLE  = 10
   NUMCYCLES     =  5
```

$!ANIMATEZONES

Syntax:

```
$!ANIMATEZONES
  START = <integer>
  END = <integer>
  [optional parameters]
```

Description: Produce an animation showing one zone at a time. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameters:

Parameter Syntax	Notes
START = *<integer>*	Starting zone number.
END = *<integer>*	Ending zone number.

Optional Parameters:

Parameter Syntax	Default	Notes
`SKIP = <integer>`	`1`	Zone skip.
`CREATEMOVIEFILE = <boolean>`	`FALSE`	If TRUE, must be preceded by $!EXPORTSETUP commands.

Example: The following example animates just the first five zones:

```
$!ANIMATEZONES
   START = 1
   END = 5
```

$!ATTACHDATASET

Syntax:

```
$!ATTACHDATASET
   [optional parameter]
```

Description: Attach the current frame to the data set of another frame. This command is usually found only in layout files generated by Tecplot. Note that the **$!FRAMEMODE** command automatically executes an **$!ATTACHDATASET** command if a frame mode is requested in a frame that does not have an attached data set. Tecplot attaches the data set from the closest frame (in drawing order) having an attached data set.

Optional Parameter:

Parameter Syntax	Default	Notes
`FRAME = <integer>`	*numframes*-1	Frames are numbered 1 to *numframes*, based on the order they are drawn when a Redraw All is executed.

Examples:

Example 1: The following example attaches to the current frame the data set from the second frame drawn when doing a Redraw All:

```
$!ATTACHDATASET
   FRAME = 2
```

Example 2: The following example attaches to the current frame the data set from the frame drawn next-to-last when doing a Redraw All:

```
$!ATTACHDATASET
```

Syntax: `$!ATTACHGEOM`
[optional parameters]
<geometryrawdata>

Description: Attach a geometry to the current frame.

Required Parameter:

Parameter Syntax	Notes
<geometryrawdata>	This is the data which defines the size and relative shape of the geometry. This must be at the end of the command after any other parameters.

Optional Parameters:

Parameter Syntax	Default	Notes
POSITIONCOORDSYS = *<coordsys>*	**GRID**	
ANCHORPOS = *<<anchorpos>>*		This assigns the anchor position of the geometry.
ZONE = *<integer>*	**1**	This is only used if **ATTACHTOZONE** = **TRUE**. This geometry is disabled if the zone assigned here is inactive.
ATTACHTOZONE = *<boolean>*	**FALSE**	If **TRUE**, must include **ZONE.**
COLOR = *<color>*	**BLACK**	
CLIPPING = *<clipping>*	**CLIPTTOVIEWPORT**	
FILLCOLOR = *<color>*	**WHITE**	
ISFILLED = *<boolean>*		
GEOMTYPE = *<geomtype>*	**LINESEGS**	
LINEPATTERN = *<linepattern>*	**SOLID**	
PATTERNLENGTH = *<dexp>*	**2%**	Set the pattern length in Y-frame units (0-100).
LINETHICKNESS = *<dexp>*	**0.1%**	Set the line thickness in Y-frame units (0-100).
NUMELLIPSEPTS = *<integer>*	**72**	Numbers of points to use when drawing ellipses and circles.
ARROWHEADSTYLE = *<arrowheadstyle>*	**PLAIN**	
ARROWHEADATTACHMENT = *<arrowheadattachment>*	**NONE**	

Parameter Syntax	Default	Notes
ARROWHEADSIZE = *<dexp>*	**5%**	Set the arrowhead size in Y-frame units (0-100).
ARROWHEADANGLE = *<dexp>*	**12**	Set the angle for arrowheads (in degrees).
SCOPE = *<scope>*	**LOCAL**	Set the scope to **GLOBAL** to draw this geometry in all "like" frames.
MACROFUNCTIONCOMMAND = *<string>*	**Null**	Set the macro command to execute when you hover over the geometry and press Ctrl-right-click. For security reasons this command can only be used in the Tecplot configuration file.
DRAWORDER= *<draworder>*	**AFTERDATA**	
IMAGEFILENAME= *<string>*		
MAINTAINASPECTRATIO= *<boolean>*	**TRUE**	
RESIZEFILTER= *<resizefilter>*	**TEXTUREFILTER**	Default = **CUBIC**

Examples:

Example 1: The following example creates a red circle, with a radius equal to 25 percent of the height of the frame, in the center of the frame:

```
$!ATTACHGEOM
   POSITIONCOORDSYS = FRAME
   ANCHORPOS
   {
    X = 50
    Y = 50
   }
   GEOMTYPE = CIRCLE
   COLOR = RED
   RAWDATA
   25
```

Example 2: The following example creates an L-shaped polyline with an arrowhead at the end:

```
$!ATTACHGEOM
   POSITIONCOORDSYS = FRAME
   ANCHORPOS
   {
    X = 20
    Y = 80
   }
```

```
GEOMTYPE = LINESEGS
ARROWHEADATTACHMENT = ATEND
RAWDATA
1
3
0 0
0 -60
40 0
```

$!ATTACHTEXT

Syntax: `$!ATTACHTEXT`
    `TEXT =` *<string>*
    [*optional parameters****]***

Description: Attach text to the current frame.

Required Parameter:

Parameter Syntax	Notes
TEXT = *<string>*	Text string to draw.

Optional Parameters:

Parameter Syntax	Default	Notes
ANCHORPOS = *<<anchorpos>>*		This assigns the anchor position for the text. Units are dependent on **POSITIONCOORDSYS.**
POSITIONCOORDSYS = *<coordsys>*	**FRAME**	
CLIPPING= *<clipping>*	**CLIPTOVIEWPORT**	
ZONE = *<integer>*	1	This is only used if **ATTACHZONE = TRUE**. This text is disabled if the zone assigned here is inactive.
ATTACHTOZONE = *<boolean>*	**FALSE**	If **TRUE**, must include **ZONE**.
COLOR = *<color>*	**BLACK**	
TEXTSHAPE **{** **FONT =** *<font>* **SIZEUNITS =** *<sizeunits>* **HEIGHT =** *<dexp>* **}**	**HELVBOLD** **POINT** **14**	The following combinations of **SIZEUNITS** and **POSITIONCOORDSYS** are allowed: **FRAME/FRAME, POINT/FRAME GRID/GRID, FRAME/GRID.**

Parameter Syntax	Default	Notes

`BOX`
`{`
` BOXTYPE = `*<boxtype>*
` LINETHICKNESS = `*<dexp>*
` MARGIN = `*<dexp>*
` COLOR = `*<color>*
` FILLCOLOR = `*<color>*
`}`
	
	`NONE`
	`0.1%`
	`20`
	`BLACK`
	`WHITE`
		The margin is the space between the text and box. The margin is measured in terms of the percentage of the text height.

`ANGLE = `*<dexp>*	`0.0`	Text angle (in degrees).
`ANCHOR = `*<textanchor>*	`LEFT`	Specifies what part of the text to anchor to the frame.
`LINESPACING = `*<dexp>*	`1.0`	Line spacing to use if text contains multiple lines.
`SCOPE = `*<scope>*	`LOCAL`	Set the scope to **GLOBAL** to include this text in all "like" frames.
`MACROFUNCTIONCOMMAND` = *<string>*	`NULL`	Set the macro command to execute when you hover over the geometry and press Ctrl-right-click. For security reasons this command can only be used in the Tecplot configuration file.

Examples:

Example 1: The following example creates the text **ABC** and positions it in the lower left corner of the frame:

```
$!ATTACHTEXT
   TEXT = "ABC"
```

Example 2: The following example creates the text TEXT AT AN ANGLE and places it in the center of the frame. The text is drawn at an angle of 45 degrees:

```
$!ATTACHTEXT
   TEXT =  "TEXT AT AN ANGLE"
   ANGLE = 45
   XYPOS {X=50 Y=50}
```

Example 3: The following example creates the text TIMES-ROMAN using the Times Roman font. This text includes a text box:

```
$!ATTACHTEXT
   TEXT =  "TIMES-ROMAN"
   FONT = TIMES
   BOX
   {
    BOXTYPE = PLAIN
    MARGIN  = 20
   }
```

```
XYPOS {X=20 Y=20}
```

$!BASICCOLOR

Syntax: `$!BASICCOLOR`
[optional parameters]

Description: A SetValue command that sets the red, green and blue components for any of the basic colors in Tecplot.

Optional Parameters:

Parameter Syntax		Notes
`BLACK`	*<<rgb>>*	
`RED`	*<<rgb>>*	
`GREEN`	*<<rgb>>*	
`BLUE`	*<<rgb>>*	
`CYAN`	*<<rgb>>*	
`YELLOW`	*<<rgb>>*	
`PURPLE`	*<<rgb>>*	
`WHITE`	*<<rgb>>*	
`CUSTOM1...CUSTOM56`	*<<rgb>>*	

Example: Set the **`CUSTOM8`** color to be brown:

```
$!BASICCOLOR
   CUSTOM8
   {
    R = 165
    G  = 42
    B  = 42
   }
```

$!BASICSIZE

Syntax: `$!BASICSIZE`

[optional parameters]

Description: A SetValue command that sets sizes of various objects like line thicknesses, line pattern length, font height, and so forth. Sizes can be assigned when interacting with Tecplot by either entering an exact value or by choosing from a preset list of values. The **$!BASICSIZE** command allows you to change the values in the preset lists.

Optional Parameters:

Parameter Syntax		Notes
LINETHICKNESSES	*<<basicsizelist>>*	
TICKLENGTHS	*<<basicsizelist>>*	
SYMBOLSIZES	*<<basicsizelist>>*	
LINEPATLENGTHS	*<<basicsizelist>>*	
ARROWHEADSIZES	*<<basicsizelist>>*	
POINTTEXTSIZES	*<<basicsizelist>>*	
FRAMETEXTSIZES	*<<basicsizelist>>*	

Example: Change the medium line pattern length to be 2.5 percent:

```
$!BASICSIZE
   LINEPATLENGTHS
   {
    MEDIUM = 2.5
   }
```

$!BLANKING

Syntax: **$!BLANKING**

[optional parameters]

Description: A SetValue command that changes settings for IJK- or value-blanking.

Optional Parameters:

Parameter Syntax	Notes
IJK **{** **INCLUDE** *<op> <boolean>* **IJKBLANKMODE** = *<ijkblankmode>* **IMINFRACT** *<op> <dexp>* **JMINFRACT** *<op> <dexp>* **KMINFRACT** *<op> <dexp>* **IMAXFRACT** *<op> <dexp>* **JMAXFRACT** *<op> <dexp>* **KMAXFRACT** *<op> <dexp>* **ZONE** = *<integer>* **}**	Minimum and maximum fractions are in terms of percentages (0-100). Zero represents an index of one and 100 the maximum index. Only one zone can be assigned to use IJK-blanking.
VALUE **{** **VALUEBLANKCELLMODE** = *<valueblankcellmode>* **BLANKENTIRECELL** = *<boolean>* **INCLUDE** = *<boolean>* **CONSTRAINT nnn** *<integer>* **{** **INCLUDE** = *<boolean>* **RELOP** = *<valueblankrelop>* **CONSTRAINTOP2MODE** = *<constraintop2mode>* **VALUECUTOFF** = *<double>* **VARA** = *<integer>* **VARB** = *<integer>* **SHOW** = *<boolean>* **COLOR** = *<color>* **LINEPATTERN** = *<linepattern>* **PATTERNLENGTH** = *<double>* **LINETHICKNESS** = *<double>* **}** **}**	Set to **FALSE** to get precision-blanking. Set to **FALSE** to turn off all value-blanking. Use *<integer>* to specify which constraint to modify.
DEPTH **{** **INCLUDE** = *<boolean>* **FROMFRONT** = *<double>* **FROMBACK** = *<double>* **}**	If TRUE, draws only those portions at the plot with depth values within the FROMFRONT and FROMBACK limits. FROMFRONT and FROMBACK are expressed as percentages of the overall 3-D depth.

Examples:

Example 1: Set IJK-blanking to cut away the minimum index corner:

```
$!BLANKING
   IJK
   {
    INCLUDE   = YES
    IMINFRACT = 0
    JMINFRACT = 0
    KMINFRACT = 0
    IMAXFRACT = 50
    JMAXFRACT = 50
    KMAXFRACT = 50
   }
```

Example 2: Use value-blanking to cut away all cells that have at least one node where variable 3 is less than or equal to 7.5:

```
$!BLANKING
   VALUE
   {
     INCLUDE = YES
     CONSTRAINT 1
     {
       INCLUDE = YES
       VARA = 3
       RELOP = LESSTHANOREQUAL
       VALUECUTOFF = 7.5
     }
   }
```

$!BRANCHCONNECTIVITY

Syntax:

```
$!BRANCHCONNECTIVITY
   ZONE    = <integer>
   [no optional parameters]
```

Description: For zones where connectivity is shared, this command allows for branching of connectivity information from the specified zone.

Required Parameters:

Parameter Syntax	Notes
ZONE = *<integer>*	

Example: Suppose Zones 2, 3 and 4 share connectivity. This command branches the connectivity of the second zone. Zones 3 and 4 will still share connectivity.

```
$!BRANCHCONNECTIVITY
   ZONE  = 2
```

$!BRANCHFIELDDATAVAR

Syntax:

```
$!BRANCHFIELDDATAVAR
   ZONE  = <integer>
   VAR   = <integer>
   [no optional parameters]
```

Description: Allows for branching of specified variable in the specified zone for zones that share variables.

Required Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	
VAR	= *<integer>*	

Example: Assume Zones 1, 2 and 4 share variables 3 and 5. This command branches the third variable from the second zone. Variable 3 will still be shared by zones 1 and 4, while variable 5 will still be shared by all three zones.:

```
$!BRANCHFIELDDATAVAR
   ZONE  = 2
   VAR   = 3
```

$!BREAK

Syntax:

```
$!BREAK
```

[no parameters]

Description: Jump out of the current **$!LOOP-ENDLOOP** or **$!WHILE-$!ENDWHILE**.

Example:

```
$!LOOP 5
   .
   .
   .
$!BREAK
   .
   .
   .
$!ENDLOOP
```

$!COLORMAP

Syntax:

```
$!COLORMAP
  [optional parameters]
```

Description: A SetValue command that changes the settings for the global contour color map and the global light source shading color map in Tecplot. Changes here affect all frames using these color maps. See **$!GLOBALCONTOUR COLORMAPFILTER** for additional settings that can be applied on a frame-by-frame basis.

Optional Parameters:

Parameter Syntax	Notes
TWOCOLOR *<<colormapcontrolpoints>>*	
SMRAINBOW *<<colormapcontrolpoint>>*	
LGRAINBOW *<<colormapcontrolpoint>>*	
MODERN *<<colormapcontrolpoints>>*	
GRAYSCALE *<<colormapcontrolpoints>>*	
USERDEFINED *<<colormapcontrolpoints>>*	
USERDEFINED NUMCONTROLPOINTS = *<int>*	
CONTOURCOLORMAP *<colormap>*	

Example: Make the third control point for the small rainbow color map to be positioned 0.44 of the way across the color map. Set the leading and trailing RGB red value to 90:

```
$!COLORMAP
   SMRAINBOW
   {
    CONTROLPOINT 3
    {
     COLORMAPFRACTION = 0.44
     LEADRGB
     {R = 90}
     TRAILRGB
     {R = 90}
    }
   }
```

$!COLORMAPCONTROL *[Required-Control Option]*

Description: The different commands in the **COLORMAPCONTROL** compound function family are described separately in the following sections.

The **COLORMAPCONTROL** compound functions are:

```
$!COLORMAPCONTROL REDISTRIBUTECONTROLPOINTS
   $!COLORMAPCONTROL COPYSTANDARD
   $!COLORMAPCONTROL RESETTOFACTORY
```

$!COLORMAPCONTROL REDISTRIBUTECONTROLPOINTS

Syntax:

```
$!COLORMAPCONTROL REDISTRIBUTECONTROLPOINTS
   [no parameters]
```

Description: Redistribute the control points for the currently active color map so they are evenly spaced across the color map. This is equivalent to clicking Redistribute Control Points in the Color Map dialog. Note that this does not change the RGB values assigned at each control point.

Example:

```
$!COLORMAPCONTROL REDISTRIBUTECONTROLPOINTS
```

$!COLORMAPCONTROL COPYSTANDARD

Syntax:

```
$!COLORMAPCONTROL COPYSTANDARD
   CONTOURCOLORMAP = <standardcolormap>
```

Description: Preset either the user-defined color map or the raw user-defined color map to be a copy of one of the standard color maps. Tecplot must currently be using either the user-defined color map or the raw user-defined color map in order to use this function.

Required Parameter:

Parameter Syntax	Notes
`CONTOURCOLORMAP = <standardcolormap>`	The color map to copy.

Example: The following example sets the current color map to be a copy of the small rainbow color map:

```
$!COLORMAPCONTROL COPYSTANDARD
   CONTOURCOLORMAP = SMRAINBOW
```

$!COLORMAPCONTROL RESETTOFACTORY

Syntax: `$!COLORMAPCONTROL RESETTOFACTORY`
    [*no parameters****]***

Description: Redistribute the control points and reset the RGB values for the currently active color map. This is equivalent to clicking Reset on the Color Map dialog.

Example:

```
$!COLORMAPCONTROL RESETTOFACTORY
```

$!COMPATIBILITY

Syntax: `$!COMPATIBILITY`
    [*optional parameters****]***

Description: Allow datasharing access and setting, without warning.

Optional Parameters:

Parameter Syntax	Default	Notes
ALLOWDATASHARING = *<boolean>*	**TRUE**	If **FALSE**, Tecplot will not allow data sharing. This may be necessary to use older add-ons that cannot handle shared data.
USEV10TEXTFORMATTING = *<boolean>*	**TRUE**	If **FALSE**, allows Tecplot to display text subscripts and superscripts created with older Tecplot versions without automatically converting the text to the new formatting.

Example: The following commands turn on datasharing:

```
$!COMPATIBILITY ALLOWDATASHARING=TRUE
```

$!CONTINUE

Syntax: `$!CONTINUE`

Description: Transfer control back to nearest `$!LOOP` or `$!WHILE`.

Example:

```
$!LOOP 10
   .
   .
   .
$!CONTINUE
   .
   .
   .
$!ENDLOOP
```

$!CONTOURLABELS *[Required-Control Option]*

Description: The different commands in the `CONTOURLABELS` compound function family are described separately in the following sections.

The `CONTOURLABELS` compound functions are:

```
$!CONTOURLABELS ADD
$!CONTOURLABELS DELETEALL
```

$!CONTOURLABELS ADD

Syntax: `$!CONTOURLABELS ADD`
    [optional parameters]

Description: Add contour labels to your plot.

Optional Parameters:

Parameter Syntax	Default	Notes

`XYZPOS`
`{`		

`X = `*<dexp>*	`0.0`	X-position for contour label.
`Y = `*<dexp>*	`0.0`	Y-position for contour label.
`Z = `*<dexp>*	`0.0`	Z-position for contour label (use Z only for 3-D plots).
`}`		

Parameter Syntax	Default	Notes
ISALIGNED = *<boolean>*	**TRUE**	If **TRUE** then align the contour label along the contour line; if **FALSE**, draw the label horizontally.
CONTOURGROUP = *<integer>*	**1**	Defines which contour group is changed.

Example: The following commands add labels at (0.5, 0.25) and (0.73, 0.17) in a 2-D field plot. The labels will be aligned:

```
$!CONTOURLABELS ADD
   CONTOURGROUP = 2
   XYZPOS
   {
    X = 0.5
    Y = 0.25
   }
$!CONTOURLABELS ADD
   XYZPOS
   {
    X = 0.73
    Y = 0.17
   }
```

$!CONTOURLABELS DELETEALL

Syntax:

```
$!CONTOURLABELS DELETEALL
   [optional parameters]
```

Description: Delete all currently defined contour labels.

Optional Parameters:

Parameter Syntax	Default	Notes
CONTOURGROUP = *<integer>*	**1**	Defines which contour group is changed.

Example:

```
$!CONTOURLABELS DELETEALL

   CONTOURGROUP = 3
```

$!CONTOURLEVELS *[Required-Control Option]*

Description: The different commands in the **CONTOURLEVELS** compound function family are described separately in the following sections.

The **CONTOURLEVELS** compound functions are:

```
$!CONTOURLEVELS ADD
$!CONTOURLEVELS NEW
$!CONTOURLEVELS DELETENEAREST
$!CONTOURLEVELS DELETERANGE
$!CONTOURLEVELS RESET
$!CONTOURLEVELS RESETTONICE
```

$!CONTOURLEVELS ADD

Syntax: **$!CONTOURLEVELS ADD**
<contourlevelrawdata>
[optional parameters]

Description: Add a new set of contour levels to the existing set of contour levels.

Required Parameter:

Parameter Syntax	Notes
<contourlevelrawdata>	Supply a list of contour levels to add.

Optional Parameters:

Parameter Syntax	Default	Notes
CONTOURGROUP = *<integer>*	1	Defines which contour group is changed.

Example: Add contour levels 1.7, 3.4 and 2.9 to the plot:

```
$!CONTOURLEVELS ADD
   RAWDATA
   3
   1.7
   3.4
   2.9
```

$!CONTOURLEVELS DELETENEAREST

Syntax:

```
$!CONTOURLEVELS DELETENEAREST
   RANGEMIN = <dexp>
   [optional parameters]
```

Description: Delete the contour level whose value is nearest the value supplied in the **RANGEMIN** parameter.

Required Parameter:

Parameter Syntax	Notes
RANGEMIN = *<dexp>*	Delete the contour level whose value is nearest to this value.

Optional Parameters:

Parameter Syntax	Default	Notes
CONTOURGROUP = *<integer>*	1	Defines which contour group is changed.

Example: Delete the contour level whose value is nearest to 3.4:

```
$!CONTOURLEVELS DELETENEAREST
   RANGEMIN = 3.4
```

$!CONTOURLEVELS DELETERANGE

Syntax:

```
$!CONTOURLEVELS DELETERANGE
   RANGEMIN = <dexp>
   RANGEMAX = <dexp>
   [optional parameters]
```

Description: Delete all contour levels between a minimum and maximum contour value (inclusive).

Required Parameters:

Parameter Syntax	Notes
`RANGEMIN = ` *<dexp>*	Minimum contour level to delete.
`RANGEMAX = ` *<dexp>*	Maximum contour level to delete.

Optional Parameters:

Parameter Syntax	Default	Notes
`CONTOURGROUP =` *<integer>*	`1`	Defines which contour group is changed.

Example: Delete all contour levels between 0.1 and 0.7:

```
$!CONTOURLEVELS DELETERANGE
   RANGEMIN = 0.1
   RANGEMAX = 0.7
```

$!CONTOURLEVELS NEW

Syntax:

```
$!CONTOURLEVELS NEW
```

<contourlevelrawdata>
[optional parameters]

Description: Replace the current set of contour levels with a new set.

Required Parameter:

Parameter Syntax	Notes
<contourlevelrawdata>	Supply a list of contour levels to add.

Optional Parameters:

Parameter Syntax	Default	Notes
`CONTOURGROUP =` *<integer>*	`1`	Defines which contour group is changed.

Example: Replace the current set of contour levels with the levels 0.5, 0.75 and 1.0:

```
$!CONTOURLEVELS NEW
```

```
RAWDATA
3
0.5
0.75
1.0
```

$!CONTOURLEVELS RESET

Syntax:

```
$!CONTOURLEVELS RESET
   NUMVALUES = <integer>
   [optional parameters]
```

Description: Reset the contour levels to a set of evenly distributed values spanning the entire range of the currently selected contouring variable.

Required Parameter:

Parameter Syntax	Notes
`NUMVALUES = <integer>`	New number of contour levels.

Optional Parameters:

Parameter Syntax	Default	Notes
`CONTOURGROUP = <integer>`	1	Defines which contour group is changed.

Example: Reset the contour levels to use 150 levels:

```
$!CONTOURLEVELS RESET
   NUMVALUES = 150
```

$!CONTOURLEVELS RESETTONICE

Syntax:

```
$!CONTOURLEVELS RESETTONICE
   APPROXNUMVALUES = <integer>
   [optional parameters]
```

Description: Reset the contour levels to a set of evenly distributed, nice values spanning the entire range of the currently selected contouring variable, with a specified number of entries.

Required Parameter:

Parameter Syntax	Notes
APPROXNUMVALUES = *<integer>*	Approximate number of contour levels desired. Actual value may be different.

Optional Parameters:

Parameter Syntax	Default	Notes
CONTOURGROUP = *<integer>*	1	Defines which contour group is changed.

Example: Reset the contour levels to use 150 levels:

```
$!CONTOURLEVELS RESETTONICE
   APPROXNUMVALUES = 10
```

$!CREATECIRCULARZONE

Syntax:

```
$!CREATECIRCULARZONE
   IMAX = <integer>
   JMAX = <integer>
 [optional parameters]
```

Description: Create a circular (or cylindrical) IJ- or IJK-ordered zone.

Required Parameters:

Parameter Syntax	Notes
IMax = *<integer>*	Radial direction.
JMax = *<integer>*	Circumferential direction, must be greater than 3.

Optional Parameters:

Parameter Syntax	Default	Notes
`KMax = `*<integer>*	`1`	Bottom to top direction
`X = `*<dexp>*	`0`	X-coordinate for center.
`Y = `*<dexp>*	`0`	Y-coordinate for center.
`Z1 = `*<dexp>*	`0`	Z-minimum if a cylinder is created.
`Z2 = `*<dexp>*	`1`	Z-maximum if a cylinder is created.
`XVAR = `*<integer>*	`Auto`	Only needed when processing journal instuctions.
`YVAR = `*<integer>*	`Auto`	Only needed when processing journal instuctions.
`ZVAR = `*<integer>*	`Auto`	Only needed when processing journal instuctions.
`RADIUS = `*<dexp>*	`1`	
`DATATYPE = `*<datatype>*	`SINGLE`	

Examples:

Example 1: Create a circular 10 by 20 IJ-ordered zone centered at (5, 5) with a radius of 2:

```
$!CREATECIRCULARZONE
   IMax     = 10
   JMax     = 20
   X        = 5
   Y        = 5
   RADIUS   = 2
```

Example 2: Create a cylindrical 5 by 6 by 8 IJK-ordered zone with the bottom centered at (4, 4, 0) and the top centered at (4, 4, 7) and a radius of 3:

```
$!CREATECIRCULARZONE
   IMax     = 5
   JMax     = 6
   KMax     = 8
   X        = 4
   Y        = 4
   Z1       = 0
   Z2       = 7
   RADIUS = 3
```

$!CREATECONTOURLINEZONES

Syntax: `$!CREATECONTOURLINEZONES`

[optional parameters]

Description: Create zones from the currently-defined contour lines. One zone can be created from each contour level in that plot, or one zone for every polyline can be generated..

Optional Parameter:

Parameter Syntax	Notes
CONTLINECREATEMODE = *[ONEZONEPERCONTOURLEVEL or ONEZONEPERINDEPENDENTPOLYLINE*	Select whether one zone per contour lever will be created or whether there will be a zone for each polyline.

Example:

```
Create a new zone for each contour line on an existing
   contour plot.
$!CREATECONTOURLINEZONES
  CONTLINECREATEMODE = ONEZONEPERCONTOURLEVEL
```

$!CREATEFEBOUNDARY

Syntax:

```
$!CREATEFEBOUNDARY
   SOURCEZONE = <integer>
   [optional parameters]
```

Description: Zone boundaries for finite element data cannot be turned on or off using the boundary plot layer in Tecplot. You can, however, create a separate zone which is the boundary of a finite element zone. This new zone can then be turned on or off. One requirement for this function to work correctly is that adjacent cells must share the same node points along their common boundary.

Required Parameter:

Parameter Syntax	Notes
SOURCEZONE = *<integer>*	Zone to extract the boundary from.

Optional Parameter:

Parameter Syntax	Default	Notes
REMOVEBLANKEDSURFACES = *<boolean>*	**FALSE**	Set to **TRUE** if you want the resulting zone to include only the boundary adjacent to non-blanked cells.

Example: Create an FE-boundary zone from zone 3:

```
$!CREATEFEBOUNDARY
   SOURCEZONE = 3
```

$!CREATEFESURFACEFROMIORDERED

Syntax:

```
$!CREATEFESURFACEFROMIORDERED
   SOURCEZONES = <set>
   [optional parameters]
```

Description: A FE-Surface zone can be generated from two or more I-Ordered zones. To get the best possible output, it is recomended that the source zones should have their nodes arranged in a similar manner so that the connecting lines between points are as straightforward as possible. For this reason, indices from source zones should increase in the same direction.

Required Parameter:

Parameter Syntax	Notes
SOURCEZONES = *<set>*	Zones whose points will be used to create the new surface.

Optional Parameter:

Parameter Syntax	Default	Notes
CONNECTSTARTTOEND = *<boolean>*	**FALSE**	**TRUE** allows for closed surfaces.

Example: Create an FE-Surface zone from zones 3 and 4:

```
$!CREATEFESURFACEFROMIORDERED
   SOURCEZONES = [3-4]
```

$!CREATEISOZONES

Syntax:

```
$!CREATEISOZONES
```

[no parameters]

Description: Create zones from the currently defined iso-surfaces. One zone will be created from each defined iso-surface. The iso-surfaces must be active and you must have at least one active volume zone.

Example:

```
$!CREATEISOZONES
```

$!CREATELINEMAP

Syntax:

```
$!CREATELINEMAP
```

[no parameters]

Description: Create a new Line-mapping.

Example:

```
$!CREATELINEMAP
```

$!CREATEMIRRORZONES

Syntax:

```
$!CREATEMIRRORZONES
   SOURCEZONES = <set>
   [optional parameters]
```

Description: Create new zones that are mirror images of the source zones

Required Parameter:

Parameter Syntax	Notes
`SOURCEZONES = <set>`	Zone(s) to create mirror zone(s) from.

Optional Parameter:

Parameter Syntax	Default	Notes
`MIRRORVAR = <mirrorvar>`	`'X'`	This variable in the new zone is multiplied by -1 after the zone is copied.

Example: Create a mirror of zones 2-4 across the Y-axis (that is, mirror the X-variable) in 2D frame mode:

```
$!CREATEMIRRORZONES
   SOURCEZONES = [2-4]
   MIRRORVAR   = 'X'
```

$!CREATENEWFRAME

Syntax:

```
$!CREATENEWFRAME
   [optional parameters]
```

Description: Creates a new frame.

Optional Parameters:

Parameter Syntax	Default	Notes

XYPOS
{
X = *<dexp>*
Y = *<dexp>*
}
	
	1.0
	0.25
		
		X-position (inches) relative to the left edge of the paper.
		Y-position (inches) relative to the top edge of the paper.

WIDTH = *<dexp>*	9	Units are in inches.
HEIGHT = *<dexp>*	8	Units are in inches.

Note: The default position and size of the initial frame created when Tecplot starts up can be changed in the Tecplot configuration file.

Example: The following example creates a 5- by 5-inch frame with the upper left hand corner of the frame positioned 2 inches from the left edge of the paper and 1 inch from the top:

```
$!CREATENEWFRAME
   XYPOS
   {
    X = 2
    Y = 1
   }
   WIDTH  = 5
   HEIGHT = 5
```

$!CREATERECTANGULARZONE

Syntax: `$!CREATERECTANGULARZONE`
[optional parameters]

Description: Create a rectangular zone. If no data set exists when this command is executed, a data set is created with variables X, Y (and Z, if *KMax > 1*). If a data set exists prior to this command, the non-coordinate variables for the zone created are initialized to zero.

Optional Parameters:

Parameter Syntax	Default	Notes
IMax = *<integer>*	1	I-dimension.
JMax = *<integer>*	1	J-dimension.
KMax = *<integer>*	1	K-dimension.
X1 = *<dexp>*	0	X-minimum.
Y1 = *<dexp>*	0	Y-minimum.
Z1 = *<dexp>*	0	Z-minimum.
X2 = *<dexp>*	1	X-maximum.
Y2 = *<dexp>*	1	Y-maximum.
Z2 = *<dexp>*	1	Z-maximum.
XVAR = *<integer>*	Auto	Only needed when processing journal instuctions.
YVAR = *<integer>*	Auto	Only needed when processing journal instuctions.
ZVAR = *<integer>*	Auto	Only needed when processing journal instuctions.
DATATYPE = *<datatype>*	SINGLE	

Example: Create a rectangular IJ-ordered zone dimensioned 20 by 30 where X ranges from 0 to 3 and Y from 3 to 9:

```
$!CREATERECTANGULARZONE
   IMax      = 20
   JMax      = 30
   X1        = 0
   Y1        = 3
   X2        = 3
   Y2       = 9
```

$!CREATESIMPLEZONE

Syntax:

```
$!CREATESIMPLEZONE
   [optional parameters]
   <xyrawdata>
```

Description: Create a new zone by specifying only a list of XY-pairs of data. If other zones exist prior to using this function and there are more than 2 variables, then the additional variables are also created and set to zero.

Required Parameter:

Parameter Syntax	Notes
<xyrawdata>	See Chapter 9 for details.

Optional Parameter:

Parameter Syntax	Default	Notes
`DATATYPE = ` *<datatype>*	`SINGLE`	

Example: Create a simple XY-zone that has the XY-pairs (1, 0), (2, 1), (3, 7) and (5 9):

```
$!CREATESIMPLEZONE
   RAWDATA
   4
   1 0
   2 1
   3 7
   5 9
```

$!CREATESLICEZONEFROMPLANE

Syntax:

```
$!CREATESLICEZONEFROMPLANE
  [optional parameters]
```

Description: Create a new zone as a slice through existing 3-D volume zones. Use **$!GLOBALTHREED** to define the slicing plane orientation.

Optional Parameters:

Parameter Syntax	Default	Notes
`SLICESOURCE=` *<slicesource>*	`VOLUMEZONES`	
`FORCEEXTRACTIONTOSINGLEZONE =` *<boolean>*	`TRUE`	

Example: Create a slice zone at *X=0*:

```
$!GLOBALTHREED
   SLICE
   {
    ORIGIN {X=0}
    NORMAL
    {
     X=1
     Y=0
     Z=0
    }
   }
   $!CREATESLICEZONEFROMPLANE
     SLICESOURCE=VOLUMEZONES
```

$!CREATESLICEZONES

Syntax:

```
$!CREATESLICEZONES
```

[no parameters]

Description: Create a new zone for each slice defined on the Slice Details dialog. Only creates slices from volume zones.

Example:

```
$!GLOBALSLICE POSITION1 {X = 6}
$!GLOBALCONTOUR VAR = 4
$!GLOBALSLICE SHOW = YES
$!GLOBALSLICE POSITION2 {X = 1}
$!GLOBALSLICE SHOWPOSITION2 = YES
$!GLOBALSLICE SHOWINTERMEDIATESLICES = YES
$!GLOBALSLICE NUMINTERMEDIATESLICES = 6
$!REDRAW
$!CREATESLICEZONES
```

$!CREATESTREAMZONES

Syntax: `$!CREATESTREAMZONES`
[optional parameters]

Description: Create one or more zones out of the currently defined streamtraces. The new zones have the same number of variables per data point as the other zones in the data set with all non-coordinate variables interpolated at the positions along the streamtrace.

Optional Parameter:

Parameter Syntax	Default	Notes
CONCATENATE = *<boolean>*	**FALSE**	Set to **TRUE** to create a single zone out of all common streamtraces. The cell that connects the end of one streamtrace with the beginning of the next can later be turned off using value-blanking.

Example: Create a single zone out of all common streamzones:

```
$!CREATESTREAMZONES
   CONCATENATE = TRUE
```

$!DATASETUP

Syntax: `$!DATASETUP`
[optional parameters]

Description: A SetValue command that sets miscellaneous parameters related to data.

Optional Parameters:

Parameter Syntax	Notes
SCRATCHDATAFIELDTYPE = *<datatype>*	Set the data type for scratch arrays used for geometries line segments and other lines. The default is **SINGLE** for Windows and **DOUBLE** for UNIX. This parameter can only be used in the Tecplot configuration file.
PREPLOTARGS = *<string>*	Arguments used to run the internal Preplot utility. The internal version of Preplot is used to convert ASCII datafiles when they are read directly into Tecplot. See Section the Tecplot User's Manual for more information on Preplot and its options.

Example: Change the arguments used to Preplot ASCII files so only zones 1, 2 and 3 are processed:

```
$!DATASETUP
   PREPLOTARGS = "-zonelist 1:3"
```

$!DEFAULTGEOM

Syntax:

```
$!DEFAULTGEOM
  [optional parameters]
```

Description: A SetValue command that sets the attributes for the default geometry. When a geometry is created interactively, its color, line thickness, and so forth, are preset based on the default geometry. This command is usually used only in the Tecplot configuration file.

Optional Parameters:

Parameter Syntax		Notes
ANCHORPOS	*<<xyz>>*	
POSITIONCOORDSYS	**=** *<coordsys>*	
SCOPE	**=** *<scope>*	
ZONE	**=** *<integer>*	
ATTACHTOZONE	**=** *<boolean>*	
COLOR	= *<color>*	
FILLCOLOR	= *<color>*	
ISFILLED	**=** *<boolean>*	
LINEPATTERN	**=** *<linepattern>*	
PATTERNLENGTH	*<op>* *<dexp>*	
LINETHICKNESS	*<op>* *<dexp>*	
NUMELLIPSEPTS	*<op>* *<integer>*	
ARROWHEADSTYLE	**=** *<arrowheadstyle>*	
ARROWHEADATTACHMENT	**=** *<arrowheadattachment>*	
ARROWHEADSIZE	*<op>* *<dexp>*	

Parameter Syntax		Notes
ARROWHEADANGLE	*<op> <dexp>*	
MACROFUNCTIONCOMMAND	= *<string>*	Set the macro command to execute when you hover over the geometry and press Ctrl-right-click.

Example: Make the default geometry line thickness 0.2 percent:

```
$!DEFAULTGEOM
   LINETHICKNESS = 0.2
```

$!DEFAULTTEXT

Syntax:

```
$!DEFAULTTEXT
   [optional parameters]
```

Description: A SetValue command that sets the attributes for the default text. When text is added to a plot interactively, its font, color, size, and so forth, are based on the default text. This command is usually used only in the Tecplot configuration file.

Optional Parameters:

Parameter Syntax		Notes
ANCHORPOS	*<<xy>>*	
POSITIONCOORDSYS	= *<coordsys>*	
SCOPE	= *<scope>*	
ZONE	*<op> <integer>*	
ATTACHTOZONE	= *<boolean>*	
CLIPPING	= *<clipping>*	
COLOR	= *<color>*	
ANGLE	*<op> <dexp>*	
ANCHOR	= *<textanchor>*	
LINESPACING	*<op> <dexp>*	
TEXTSHAPE	*<<textshape>>*	
BOX	*<<textbox>>*	
MACROFUNCTIONCOMMAND	= <string>	Set the macro command to execute when you hover over the geometry and press Ctrl-right-click.

Example: Make the default text font **TIMESBOLD** with a character height of 14 points:

```
$!DEFAULTTEXT
   TEXTSHAPE
   {
    FONT = TIMESBOLD
    SIZEUNITS = POINT
    HEIGHT = 14
   }
```

$!DELAY

Syntax:

```
$!DELAY <integer>
   [no parameters]
```

Description: Delay Tecplot execution for *<integer>* seconds.

Example: Pause Tecplot for 3 seconds:

```
$!DELAY 3
```

$!DELETEAUXDATA

Syntax:

```
$!DELETEAUXDATA
   AUXDATALOCATION   = [zone/dataset/frame]
   [optional parameters]
```

Description: Delete Auxilary Data in the form of name/value pairs from zones, frames or datasets.

Required Parameters:

Parameter Syntax	Notes
AUXDATALOCATION = *<zone/dataset/frame>*	Options are ZONE, DATASET or FRAME

Optional Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	Only required if A**UXDATALOCATION = zone**
NAME	= *<string>*	

Example: Delete the selected Auxilary Data from Zone 2.:

```
$!DELETEAUXDATA
   AUXDATALOCATION = zone
   ZONE = 2
   NAME = VARIABLE DATA
```

$!DELETELINEMAPS

Syntax: **$!DELETEMAPS** *<set>*
[no parameters]

Description: Delete one or more Line-mappings. If *<set>* is omitted then all Line-mappings are deleted.

Example: Delete Line-mappings 2, 3, 4 and 8:

```
$!DELETELINEMAPS [2-4,8]
```

$!DELETEVARS

Syntax: **$!DELETEVARS** *<set>*
[no parameters]

Description: Delete one or more variables.

Example: Delete variables 4 and 10:

```
$!DELETEZONES [4,10]
```

$!DELETEZONES

Syntax: `$!DELETEZONES` *<set>*
[no parameters]

Description: Delete one or more zones.

Example: Delete zones 3, 7, 8, 9 and 11:

```
$!DELETEZONES [3,7-9,11]
```

$!DOUBLEBUFFER *[Required-Control Option]*

Description: The different commands in the **`DOUBLEBUFFER`** compound function family are described separately in the following sections.

The **`DOUBLEBUFFER`** compound functions are:

```
$!DOUBLEBUFFER OFF
   $!DOUBLEBUFFER ON
   $!DOUBLEBUFFER SWAP
```

$!DOUBLEBUFFER OFF

Syntax: `$!DOUBLEBUFFER OFF`
[no parameters]

Description: Turn off double buffering; use this command once at the end of a sequence of using the double buffer.

Example: `See $!DOUBLEBUFFER SWAP`

$!DOUBLEBUFFER ON

Syntax: `$!DOUBLEBUFFER ON`
[no parameters]

Description: Turn on double buffering; use this command once at the beginning of a sequence of using the double buffer. While double buffering is turned on all drawing is sent to the back buffer.

Example: See **$!DOUBLEBUFFER SWAP**

$!DOUBLEBUFFER SWAP

Syntax:

```
$!DOUBLEBUFFER SWAP
```

[no parameters]

Description: Swap the back buffer to the front. In other words, copy the image in the back buffer to the front.

Example: The following example uses the double buffer to show the rotation of a 3-D object:

```
$!DOUBLEBUFFER ON
   $!LOOP 10
   $!ROTATE3DVIEW X
     ANGLE = 5
   $!REDRAW
   $!DOUBLEBUFFER SWAP
   $!ENDLOOP
   $!DOUBLEBUFFER OFF
```

$!DRAWGRAPHICS

Syntax: **$!DRAWGRAPHICS** *<boolean>*

[no parameters]

Description: Turn on or off all graphics drawing. Turning off all graphics during preliminary portions of a macro file can greatly increase the efficiency of the macro.

Example: Turn off all graphics drawing:

```
$!DRAWGRAPHICS NO
```

$!DROPDIALOG

Syntax: **$!DROPDIALOG** *<dialogname>*

[no parameters]

Description: Drop a Tecplot interface dialog when *<dialogname>* can be one of ADVANCED3DCONTROL, AXISEDIT, COLORMAP, CONTOUR, CREATE1DLINE, CREATECIRCULARZONE, CREATERECTANGULARZONE, CREATEZONEFROMPOLYLINES, CREATEZONEFROMVALUES, CURVEINFO, DATAINFO, DATALABELS, DATASPREADSHEET, DELETEVARIABLES, DELETEZONES, DEPTHBLANKING, DUPLICATEZONE, EQUATION, EXPORT, EXTRACTCONTOURLINES, EXTRACTDISCRETEPOINTS, EXTRACTFEBOUNDARY, EXTRACTISOSURFACES, EXTRACTPOINTSFROMGEOMETRY, EXTRACTPOINTSFROMPOLYLINE, EXTRACTSLICEFROMPLANE, EXTRACTSLICES, EXTRACTSTREAMTRACES, EXTRACTSUBZONE, IJKBLANKING, IMPORT, INVERSEDISTANCEINTERPOLATION, ISOSURFACES, KRIGINGINTERPOLATION, LIGHTSOURCE, LINEARINTERPOLATION, LINEMAPLEGEND, LOADDATA, MACROPLAY, MACRORECORD, MACROVIEWER, MIRRORZONE, NEWLAYOUT, OPENLAYOUT, ORDERFRAMES, PAPERSETUP, POLARDRAWINGOPTIONS, PRINT, PROBEAT, PROBE, QUICKEDIT, QUICKMACROPANEL, RESET3DAXES, RGBCOLORLEGEND, RGBCOLORVARSANDRANGE, ROTATE2DDATA, RULERGRID, SAVEAS, SAVE, SCATTERLEGEND, SCATTERREFERENCESYMBOL, SCATTERSIZEANDFONT, SLICES, SMOOTH, SPATIALVARS, STREAMTRACES, STYLELINKING, THREEDAXISLIMITS, THREEDORIENTATIONAXIS, THREEDVIEWDETAILS, THREEDVIEWROTATE, TRANSFORMCOORDINATES, TRANSLATEMAGNIFY, TRIANGULATE, TWODDRAWORDER, VALUEBLANKING, VECTORARROWHEADS, VECTORLENGTH, VECTORREFERENCEVECTOR, VECTORVARS, WRITEDATA, ZONEMAPSTYLE. This command is mainly useful for the Tecplot demo. To launch a dialog use **`$!LAUNCHDIALOG`**.

Example:

```
$!DROPDIALOG MACROVIEWER
```

$!DUPLICATELINEMAP

Syntax:

```
$!DUPLICATELINEMAP
   SOURCEMAP = <integer>
   DESTINATIONMAP = <integer>
```

Description: Copy attributes from an existing Line-mapping to another.

Required Parameters:

Parameter Syntax	Notes
SOURCEMAP = *<integer>*	Line-mapping from which to copy.
DESTINATIONMAP = *<integer>*	The destination can either be the number of an existing map or 1 greater than the current number of maps. If you choose the latter, a new Line-mapping will be created.

Example: Copy attributes of Line-mapping 3 to Line-mapping 7:

```
$!DUPLICATELINEMAP
   SOURCEMAP      = 3
   DESTINATIONMAP = 7
```

$!DUPLICATEZONE

Syntax:

```
$!DUPLICATEZONE
   SOURCEZONE = <integer>
   [optional parameters]
```

Description: Make a copy of an existing zone. You can assign index ranges to create a new zone which is a subset of the source zone.

Required Parameter:

Parameters Syntax	Notes
`SOURCEZONE = ` *<integer>*	Zone to duplicate (the source zone).

Optional Parameters:

Parameters Syntax	Default	Notes

`IRANGE`
`{`
`MIN = ` *<integer>*
`MAX = ` *<integer>*
`SKIP = ` *<integer>*
`}`
	
	`1`
	`0`
	`1`
		See notes on index ranges for **`$!ALTERDATA`** action command.

`JRANGE`
`{`
`MIN = ` *<integer>*
`MAX = ` *<integer>*
`SKIP = ` *<integer>*
`}`
	
	`1`
	`0`
	`1`
		See notes on index ranges for **`$!ALTERDATA`** action command.

`KRANGE`
`{`
`MIN = ` *<integer>*
`MAX = ` *<integer>*
`SKIP = ` *<integer>*
`}`
	
	`1`
	`0`
	`1`
		See notes on index ranges for **`$!ALTERDATA`** action command.

Examples:

Example 1: Make a complete copy of zone 2:

```
$!DUPLICATEZONE
   SOURCEZONE = 2
```

Example 2: Duplicate zone 3 creating a zone which uses only the I-index range from 2 to 7 from the source zone:

```
$!DUPLICATEZONE
   SOURCEZONE = 3
   IRANGE
   {
    MIN  = 2
    MAX  = 7
   }
```

$!ELSE

Syntax: `$!ELSE`

[no parameters]

Description: Conditionally handle macro commands. Used when an `$!IF` statement is `FALSE`.

Example:

```
$!VARSET |C| = 2
$!IF |C| == 5
  $!CREATENEWFRAME
   XYPOS
                {
                X = 2.5
                Y = 1.5
                }
                WIDTH  = 4
                HEIGHT = 4
   $!ELSE
    $!CREATENEWFRAME
                XYPOS
                {
                X = 3
                Y = 2
                }
                WIDTH  = 3
                HEIGHT = 3
$!ENDIF
```

$!ELSEIF

Syntax: **`$!ELSEIF`** *<conditionalexp>*

Description: Conditionally handle macro commands. Used to create multiple options for statements should an **`$!IF`** statement be **`FALSE`**.

Example:

```
$!VARSET |A| = 2
$!IF |A| < 5
  $!CREATENEWFRAME
                XYPOS
                {
                X = 1
                Y = 1
                }
                WIDTH  = 3
                HEIGHT = 3
$!ELSEIF |A| > 5
  $!CREATENEWFRAME
                XYPOS
                {
                X = 2
                Y = 1
                }
                WIDTH  = 5
                HEIGHT = 5
$!ELSE
  $!CREATENEWFRAME
                XYPOS
                {
                X = 3
                Y = 3
                }
                WIDTH  = 9
                HEIGHT = 9
$!ENDIF
```

$!EXPORT

Syntax: **`$!EXPORT`**
[no parameters]

Description: Export an image file from Tecplot. See the **`$!EXPORTSETUP`** command for details on setting up the exported image type. The **`$!EXPORT`** command is not valid for animation formats. (AVI and Raster Metafile.)

Example:

```
$!EXPORTSETUP EXPORTFORMAT = PNG
   $!EXPORT
```

$!EXPORTCANCEL

Syntax:

```
$!EXPORTCANCEL
```

[no parameters]

Description: Cancel out of the current export animation sequence. The animation file being generated is removed.

Example:

```
$!EXPORTCANCEL
```

$!EXPORTFINISH

Syntax:

```
$!EXPORTFINISH
```

[no parameters]

Description: Signals the completion of an animation sequence and causes the animation file to be created. You must call **$!EXPORTSTART** prior to using **$!EXPORTFINISH**. This command is only valid for animation formats. (AVI and Raster Metafile.) You may use the |**EXPORTISRECORDING**| intrinsic variable to make sure that an animation sequence has been initiated.

Example:

```
$!EXPORTSETUP
     EXPORTFNAME="rotate.avi"
     EXPORTFORMAT=AVI
   $!EXPORTSTART
   $!LOOP 5
   $!ROTATE3DVIEW X
     ANGLE=5
   $!EXPORTNEXTFRAME
   $!ENDLOOP
   $!IF "|EXPORTISRECORDING|" =="YES"
    $!EXPORTFINISH
$!ENDIF
```

$!EXPORTNEXTFRAME

Syntax: **$!EXPORTNEXTFRAME**
[no parameters]

Description: Records the next frame of an animation. You must call **$!EXPORTSTART** prior to calling **$!EXPORTNEXTFRAME**. This command is only valid for animation formats. (AVI and Raster Metafile. You may use the |**EXPORTISRECORDING**| intrinsic variable to make sure that an animation sequence has been initiated.)

Example:

```
$!EXPORTSETUP
     EXPORTFNAME="rotate.avi"
     EXPORTFORMAT=AVI
   $!EXPORTSTART
   $!LOOP 5
   $!ROTATE3DVIEW X
     ANGLE=5
   $!EXPORTNEXTFRAME
   $!ENDLOOP
   $!EXPORTFINISH
```

$!EXPORTSETUP

Syntax: **$!EXPORTSETUP**
[optional parameters]

Description: A SetValue command that sets the attributes for exporting image files from Tecplot. Exporting is usually intended as a means to transfer images from Tecplot to be imported by other applications. See **$!PRINTSETUP** and **$!PRINT** for generating output intended for printers and plotters.

Optional Parameters:

Parameter Syntax		Notes
EXPORTFNAME	**=** *<string>*	
EXPORTFORMAT	**=** *<exportformat>*	
GRAYSCALEDEPTH	= *<integer>*	Valid values are 0, 1, 4, 8.
IMAGEWIDTH	*<op>* *<integer>*	
SUNRASTERFORMAT	**=** *<sunrasterformat>*	Only applies if **EXPORTFORMAT** is **SUNRASTER.**

Parameter Syntax		Notes
BITDUMPREGION	= *<bitdumpregion>*	
EPSPREVIEWIMAGE **{** **IMAGETYPE** **IMAGEWIDTH** **IMAGEHEIGHT** **GRAYSCALEDEPTH** **}**	= *<epspreviewimagetype>* = *<integer>* = *<integer>* = *<integer>*	Valid values are 0, 1, 4, 8.
CONVERTTO256COLORS	= *<boolean>*	Used for TIFF, BMP, and PNG formats.
ANIMATIONSPEED	= *<double>*	Applies to AVI only. Sets the animation speed in frames per second.
USEMULTIPLECOLORTABLES	= *<boolean>*	Applies to AVI and Raster Metafile only.
TIFFBYTEORDER	= *<tiffbyteorder>*	
QUALITY	= *<integer>*	Range is from 1-100
JPEGENCODING	= **STANDARD** *or* **PROGRESSIVE**	
USESUPERSAMPLEANTIALIASING	= *<boolean>*	Default = **FALSE**
SUPERSAMPLEFACTOR	= *<integer>*	Default = 3. This is the factor used in antialiasing while reducing the size of an exported image. A larger size can improve the quality of the image, but slows performance.
PRINTRENDERTYPE	= *<printrendertype>*	Default = **PRINTRENDERTYPE_VECTOR**
RESIZEFILTER	= *<resizefilter>*	Default = **CUBICFILTER**. **TEXTURE FILTER** and **BOXFILTER** not allowed.

Example: Set up Tecplot to export a Raster Metafile image to the file **movie.rm**:

```
$!EXPORTSETUP
   EXPORTFNAME = "movie.rm"
   EXPORTFORMAT = RASTERMETAFILE
```

$!EXPORTSTART

Syntax: **$!EXPORTSTART**
[no parameters]

Description: Signals the start of an animation sequence and records the first frame of the animation. This command is only valid for animation formats. (AVI and Raster

Metafile.)

Example:

```
$!EXPORTSETUP
    EXPORTFNAME="rotate.avi"
    EXPORTFORMAT=AVI
  $!EXPORTSTART
  $!LOOP 5
  $!ROTATE3DVIEW X
    ANGLE=5
  $!EXPORTNEXTFRAME
  $!ENDLOOP
  $!EXPORTFINISH
```

$!EXTRACTFROMGEOM

Syntax:

```
$!EXTRACTFROMGEOM
  [optional parameters]
```

Description: Extract data from a 2- or 3-D field plot. The locations at which to extract the data come from a polyline geometry that must be picked prior to issuing this command.

Optional Parameters

Parameters Syntax	Default	Notes
EXTRACTLINEPOINTSONLY = *<boolean>*	**FALSE**	If **FALSE**, must include **NUMPTS**.
INCLUDEDISTANCEVAR = *<boolean>*	**FALSE**	If **TRUE,** then Tecplot includes an extra variable in the result which is the distance along the line of points extracted and **EXTRACTTOFILE** must also be **TRUE**.
NUMPTS = *<integer>*	---	Required if **EXTRACTLINEPOINTSONLY** is **FALSE**.
EXTRACTTOFILE = *<boolean>*	**FALSE**	If **FALSE**, a zone is created. If **TRUE**, must include **FNAME.**
FNAME = *<string>*	---	File name for extracted file. Required if **EXTRACTTOFILE** is **TRUE**.

Example: Extract 20 points from along the currently picked geometry. Send the result to a file called `extract.dat`:

```
$!EXTRACTFROMGEOM
  NUMPTS                    = 20
```

```
    EXTRACTTOFILE          = TRUE
    FNAME            = "extract.dat"
```

$!EXTRACTFROMPOLYLINE

Syntax: **$!EXTRACTFROMPOLYLINE**
[optional parameters]
<xyzrawdata>

Description: Extract data from a 2- or 3-D field plot. The locations of where to extract the data from come from a supplied polyline in the form of *<xyzrawdata>*.

Optional Parameters

Parameters Syntax	Default	Notes
EXTRACTTHROUGHVOLUME = *<boolean>*	**FALSE**	If **TRUE,** data is extracted from XYZ-coordinates in the polyline. If **FALSE,** data is extracted from the surface.
EXTRACTLINEPOINTSONLY = *<boolean>*	**FALSE**	If **FALSE**, must include **NUMPTS.**
INCLUDEDISTANCEVAR = *<boolean>*	**FALSE**	If **TRUE,** Tecplot includes an extra variable in the result which is the distance along the line of points extracted and **EXTRACTOFILE** must also be **TRUE**.
NUMPTS = *<integer>*	---	Required if **EXTRACTLINEPOINTSONLY** is **FALSE.**
EXTRACTTOFILE = *<boolean>*	**FALSE**	If **FALSE**, a zone is created. If **TRUE**, you must include **FNAME.**
FNAME = *<string>*	---	File name for extracted file. Required if **EXTRACTTOFILE** is **TRUE**.

Example: Extract 10 points from specific locations in a field plot. Create a zone with the extracted data:

```
$!EXTRACTFROMPOLYLINE
   EXTRACTLINEPOINTSONLY = TRUE
   RAWDATA
   10
   0 0 0
   1 2 0
   2 4 0
   3 2 0
   3 4 0
   4 4 0
```

```
4 5 0
4 6 0
5 7 0
6 9 0
```

$!FIELD

Syntax: **`$!FIELD`** *[<set>]*
[optional parameters]

Description: A SetValue command that assigns zone attributes for field plots. The <*set*> parameter immediately following the **`$!FIELD`** command is optional. If <*set*> is omitted then the assignment is applied to all zones. Otherwise the assignment is applied only to the zones specified in <*set*>.

Optional Parameters:

Parameter Syntax	Notes
MESH **{** **SHOW** = *<boolean>* **MESHTYPE** = *<meshplottype>* **COLOR** = *<color>* **LINEPATTERN** = *<linepattern>* **PATTERNLENGTH** *<op> <dexp>* **LINETHICKNESS** *<op> <dexp>* **}**	
CONTOUR **{** **SHOW** = *<boolean>* **CONTOURTYPE** = *<meshplottype>* **COLOR** = *<color>* **LINEPATTERN** = *<linepattern>* **PATTERNLENGTH** *<op> <dexp>* **LINETHICKNESS** *<op> <dexp>* **USELIGHTINGEFFECT** = *<boolean>* **FLOODCOLORING** = *<contourcoloring_e>* **LINECONTOURGROUP** = *<sminteger_t>* **}**	

Parameter Syntax	Notes
VECTOR **{** **SHOW** = *<boolean>* **VECTORTYPE** = *<vectorplottype>* **ARROWHEADSTYLE** = *<arrowheadstyle>* **COLOR** = *<color>* **ISTANGENT** = *<boolean>* **LINEPATTERN** = *<linepattern>* **PATTERNLENGTH** = *<dexp>* **LINETHICKNESS** = *<dexp>* **}**	
SCATTER **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **FILLMODE** = *<fillmode>* **FILLCOLOR** = *<color>* **SIZEBYVARIABLE** = *<boolean>* **FRAMESIZE** *<op> <dexp>* **LINETHICKNESS** *<op> <dexp>* **SYMBOLSHAPE** *<<symbolshape>>* **}**	Scatter sizing variable must be defined before this can be set to **TRUE**. See the **$!GLOBALSCATTER** command. Size of symbols when **SIZEBYVARIABLE is FALSE**.
POINTS **{** **IJKSKIP** *<<ijk>>* **POINTSTOPLOT** *<pointstoplot>* **}**	Limits the number of vectors or scatter symbols drawn.
SHADE **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **USELIGHTINGEFFECT** = *<boolean>* **}**	
BOUNDARY **{** **SHOW** = *<boolean>* **IBOUNDARY** = *<boundarysetting>* **JBOUNDARY** = *<boundarysetting>* **KBOUNDARY** = *<boundarysetting>* **COLOR** = *<color>* **LINETHICKNESS** = *<dexp>* **}**	Applies for IJ-, IK-, and IJK-ordered zones. Applies for IJ-, JK-, and IJK-ordered zones. Applies for IK-, JK-, and IJK-ordered zones.
SURFACEEFFECTS **{** **SURFACETRANSLUCENCY** = *<translucency>* **USETRANSLUCENCY** = *<boolean>* **LIGHTINGEFFECT** = *<lightingeffect>* **}**	When reading in older layouts, **FLOODTRANSLUCENCY** is ignored if SHADE layer is on for that zone, otherwise it is converted to **SURFACETRANSLUCENCY**. In a macro, this is ignored. **SURFACETRANSLUCENCY** range is one to 99.

Parameter Syntax		Notes

`SURFACES`
`{`
` SURFACESTOPLOT`
` IRANGE`
` JRANGE`
` KRANGE`
`}`
	
	= *<surfacestoplot>*
	= *<<indexrange>>*
	= *<<indexrange>>*
	= *<<indexrange>>*
		VOLUMEMODE applies to volume zones, with the exception that **POINTSTOPLOT** also applies to finite-element surface zones.

`VOLUMEMODE`
`{`
` VOLUMEOBJECTSTOPLOT`
`}`
	
	= *<<volumeobjectstoplot>>*	

`GROUP`	= *<integer>*	

Examples:

Example 1: Change the contour plot type to flood for zones 1-12:

```
$!FIELD [1-12]
   CONTOUR
   {
    CONTOURTYPE = FLOOD
   }
```

Example 2: Change the mesh color to red for all zones:

```
$!FIELD
   MESH
   {
    COLOR = RED
   }
```

$!FIELDLAYERS

Syntax:

```
$!FIELDLAYERS
   [optional parameters]
```

Description: A SetValue command that turns field plot layers on or off, or sets the 2-D draw order.

Optional Parameters:

Parameter Syntax	Notes
SHOWMESH = *<boolean>*	
SHOWCONTOUR = *<boolean>*	
SHOWVECTOR = *<boolean>*	Vector variables must be defined. See **$!GLOBALTWODVECTOR** or **$!GLOBALTHREEDVECTOR.**
SHOWSCATTER = *<boolean>*	
SHOWSHADE = *<boolean>*	
SHOWBOUNDARY = *<boolean>*	
TWODDRAWORDER = *<twoddraworder>*	
USETRANSLUCENCY = *<boolean>*	
USELIGHTINGEFFECT = *<boolean>*	

Example: Turn on the scatter layer:

```
$!FIELDLAYERS
   SHOWSCATTER = YES
```

$!FILECONFIG

Syntax:

```
$!FILECONFIG
  [optional parameters]
```

Description: A SetValue command that sets file path information in Tecplot.

Optional Parameters:

Parameter Syntax		Notes
DATAFILEVARLOADMODE	= *<varloadmode>*	Set the default loading mode for variables. The default is **BYNAME**. To get Tecplot Version 7.0 behavior, use **BYPOSITION**.
LAYOUTCONFIG **{**		
USERELATIVEPATHS	= *<boolean>*	If **TRUE**, files will be referenced using relative pathis in layout files.
INCLUDEDATA	= *<boolean>*	Default set to **TRUE** to make option to save layout packages the default.
INCLUDEPREVIEW **}**	= *<boolean>*	If **TRUE**, option to include preview image in layout packages is turned on by default.

Parameter Syntax	Notes
TEMPFILEPATH = *<string>*	Set the directory where you want Tecplot to store temporary files.

FNAMEFILTER
{
OUTPUTLAYOUTFILE = *<string>*
OUTPUTLAYOUTPACKAGEFILE = *<string>*
INPUTDATAFILE = *<string>*
OUTPUTASCIIDATAFILE = *<string>*
OUTPUTBINARYDATAFILE = *<string>*
INPUTLAYOUTFILE = *<string>*
STYLEFILE = *<string>*
MACROFILE = *<string>*
EQUATIONFILE = *<string>*
COLORMAPFILE = *<string>*
IMPORTIMAGEFILE = *<string>*
}
= *<string>*
	Default extension for saving linked layout files.
	Default extension for saving layout package files.
	Default extension for Tecplot input data files.
	Default extension for ASCII output data files.
	Default extension for binary output data files.
	Default extension for loading layout files.
	Default extension for style files.
	Default extension for macro files.
	Default extension for equation files.
	Default extension for color map files.
	Default extension for image files.

DOAUTOFNAMEEXTENSION = *<boolean>*	
DOAUTOFNAMEEXTENSIONWARNING = *<boolean>*	If TRUE a warning is displayed when attempting to save with an extension other than the default extension.

File Name Filters: Valid chracters are upper or lowercase A-Z, and 0-9. Each filter should be preceded by (*.). or it will not filter properly. On Windows, to allow more than one extension, separate them with a semicolon (;). On UNIX multiple extensions will not filter correctly unless they follow the standard UNIX shell filter format.

Windows Example: This example filters all four extensions when opening a layout file.

```
$!FILECONFIG FNAMEFILTER {INPUTLAYOUTFILE =
   "*.wsf;*.dwr;*.lay;*.lpk"}
```

Windows Example: This example filters both extensions when writing a layout file. The default extension is **.wsf** because it is the first extension presented in the list.

```
$!FILECONFIG FNAMEFILTER {OUPUTLAYOUTFILE = "
   .wsf;*.lay"}
```

Motif Example: This example filters **.aek**, **.plt**, and more.

```
$!FILECONFIG FNAMEFILTER {INPUTDATAFILE = "
   *.[ae][el][kt]"}
```

Motif Example: This example filters **.dat**, **.cam**, and more. The default extension is **.dat** because D and T are the first letters presented within the brackets.

```
$!FILECONFIG FNAMEFILTER {OUTPUTASCIIDATAFILE =
   "*.[dc]a[tm]"}
```

Example: Set the directory where Tecplot stores temporary files to be **/usr/tmp**:

```
$!FILECONFIG
   DATAFILEVARLOADMODE = BYPOSITION
    TEMPFILEPATH = "/usr/tmp"
    LAYOUTCONFIG {USERELATIVEPATHS = TRUE}
    FNAMEFILTER
       {
       INPUTDATAFILE = "*.[pd][la]t"
       COLORMAPFILE = "*.clr"
       }
```

$!FONTADJUST

Syntax:

```
$!FONTADJUST
   [optional parameters]
```

Description: A SetValue command that sets character spacing and sizing for fonts in Tecplot. These parameters are rarely changed.

Optional Parameters:

Parameter Syntax			Notes
INTERCHARSPACING	*<op>*	*<integer>*	Increase or decrease intercharacter spacing. Units are in pixels on the screen.
SUBSUPFRACTION	*<op>*	*<double>*	Size of subscript and superscript characters relative to the font height.
BOLDFACTOR	*<op>*	*<double>*	Thickness of bold characters relative to normal.
STROKEFONTLINETHICKNESS	*<op>*	*<double>*	Thickness (in frame units) of lines used to draw stroke fonts.

Example: Make superscript and subscript characters 1/3 the font height:

```
$!FONTADJUST
   SUBSUPFRACTION = 0.333
```

$!FRAMECONTROL *[Required-Control Option]*

Description: The different commands in the **FRAMECONTROL** compound function family are described separately in the following sections.

The **FRAMECONTROL** compound functions are:

```
$!FRAMECONTROL DELETETOP
   $!FRAMECONTROL FITALLTOPAPER
   $!FRAMECONTROL POP
   $!FRAMECONTROL POPATPOSITION
   $!FRAMECONTROL PUSHTOP
   $!FRAMECONTROL POPBYNAME
   $!FRAMECONTROL PUSHBYNAME
```

$!FRAMECONTROL DELETETOP

Syntax: `$!FRAMECONTROL DELETETOP`
[no parameters]

Description: Delete the top (active) frame. If there is only one frame when this is called, a new empty frame is automatically created after this command is executed. (Thus, you can never have a workspace without at least one frame.)

Example:

```
$!FRAMECONTROL DELETETOP
```

$!FRAMECONTROL FITALLTOPAPER

Syntax: `$!FRAMECONTROL FITALLTOPAPER`
[no parameters]

Description: Resize all frames so that they fit inside the hardclip limits of the paper.

Example:

```
$!FRAMECONTROL FITALLTOPAPER
```

$!FRAMECONTROL POP

Syntax: `$!FRAMECONTROL POP`
[optional parameters]

Description: Pop a frame to the top (make it the active frame).

Optional Parameter:

Parameter Syntax	Default	Notes
`FRAME = ` *<integer>*	`1`	Frame to be popped. Frames are numbered 1 to *numframes* with frame 1 drawn first when a Redraw All is executed and the highest numbered frame drawn last.

Example: Pop frame number 2:

```
$!FRAMECONTROL POP
   FRAME = 2
```

$!FRAMECONTROL POPATPOSITION

Syntax:

```
$!FRAMECONTROL POPATPOSITION
   X = <dexp>
   Y = <dexp>
```

Description: Pop the top most frame at a specified position on the paper.

Required Parameters:

Parameter Syntax	Notes
`X = ` *<dexp>*	X is in inches from the left edge of the paper.
`Y = `*<dexp>*	Y is in inches from the top edge of the paper.

Example: Pop the frame beneath the location 2 inches from the top edge of the paper and 3 inches from the left edge of the paper:

```
$!FRAMECONTROL POPATPOSITION
   X = 3
   Y = 2
```

$!FRAMECONTROL POPBYNAME

Syntax:

```
$!FRAMECONTROL POPBYNAME
   NAME = <string>
```

Description: Pop the specified frame to the top of the view stack.

Example:

```
$!FRAMECONTROL POPBYNAME
   NAME = "BANANA"
```

$!FRAMECONTROL PUSH

Syntax:

```
$!FRAMECONTROL PUSH
```
[optional parameters]

Description: Push a frame to the bottom of the frame stack (it is given the frame number 1 and therefore drawn first).

Optional Parameter:

Parameter Syntax	Default	Notes
FRAME = <*integer*>	*numframes*	Frame to be pushed. Frames are numbered 1 to *numframes* with frame 1 drawn first and the highest numbered frame drawn last when a Redraw All is executed.

$!FRAMECONTROL PUSHBYNAME

Syntax:

```
$!FRAMECONTROL PUSHBYNAME
   NAME = <string>
```

Description: Push the specified frame to the bottom of the view stack.

Example:

```
$!FRAMECONTROL PUSHBYNAME
   NAME = "BANANA"
```

$!FRAMECONTROL PUSHTOP

Syntax:

```
$!FRAMECONTROL PUSHTOP
```
[no parameters]

Description: Push the top (active) frame to the bottom.

Example:

```
$!FRAMECONTROL PUSHTOP
```

Syntax: `$!FRAMELAYOUT`
[optional parameters]

Description: A SetValue command that sets the position, border, and background attributes for the current frame. Use the **`$!FRAMECONTROL`** action command to push and pop frames if you want to change the settings for a frame other than the current frame.

Optional Parameters:

Parameter Syntax		Notes
SHOWBORDER	= *<boolean>*	
SHOWHEADER	= *<boolean>*	
ISTRANSPARENT	= *<boolean>*	
BACKGROUNDCOLOR	= *<color>*	Only applies if **ISTRANSPARENT** = **FALSE**.
HEADERCOLOR	= *<color>*	Only applies if **SHOWHEADER** = **TRUE.**
HEADERFONT	= *<font>*	
BORDERTHICKNESS	*<op> <dexp>*	Value is in Y-frame units.
WIDTH	*<op> <dexp>*	Value is in inches.
HEIGHT	*<op> <dexp>*	Value is in inches.
XYPOS	*<<xy>>*	Position of upper left corner of the frame in inches from left and top edge of the paper.

Example: Place the current frame in the upper left corner of the paper (offset 0.5 inches from the top and left edges), make the frame dimensions 3 by 4 inches, and turn off the frame border:

```
$!FRAMELAYOUT
   SHOWBORDER = NO
   XYPOS
   {
    X = 0.5
    Y = 0.5
   }
   WIDTH = 3
   HEIGHT = 4
```

$!FRAMENAME

Syntax: **$!FRAMENAME =** *<string>*
[no parameters]

Description: Set the name for the current frame.

Example: **$!FRAMENAME = "Pressure Contours for well 33"**

$!FRAMESETUP

Syntax: **$!FRAMESETUP**
[optional parameters]

Description: A SetValue command that sets parameters used to preset dynamic frame attributes when a frame is initialized.

Optional Parameters:

Parameter Syntax	Notes
ALIGNINGCONTOURLABELS = *<boolean>*	If **TRUE**, the next interactively placed contour label is aligned to the contour line.
VECTMINLEN *<op> <dexp>*	Minimum length in centimeters. Vectors shorter than this length are not drawn.
VECTDEFLEN *<op> <dexp>*	When a vector plot is drawn for the first time the vector magnitude is adjusted so the longest vector is **VECTDEFLEN** units long. **VECDEFLEN** is in frame units.
INITIAL3DSCALE *<op> <dexp>*	Initial scale for 3-D plots.
NUMSTREAMRAKEPOINTS *<op> <integer>*	Number of points to place along streamtrace rakes.

Example: Make the default length for the longest vector five percent:

```
$!FRAMESETUP
   VECTDEFLEN = 5
```

$!GETAUXDATA

Syntax: **$!GETAUXDATA** *<macrovar>*
    AUXDATALOCATION = *[zone/var/dataset/frame/linemap]*
    NAME = *<string>*
    [optional parameters]

Description: Retrieve Auxiliary Data in the form of name/value pairs and save it to the macrovariable.

Required Parameters:

Parameter Syntax		Notes
AUXDATALOCATION	= *zone/dataset/frame*	
NAME	= *<string>*	Name of existing auxiliary data

Optional Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	Only required if **AUXDATALOCATION** = zone
VAR	= *<integer>*	Only required if **AUXDATALOCATION** = var
		Only required if **AUXDATALOCATION** = linemap

Example: Get the Auxiliary Data from Zone 2, and store it in the macro variable |**ABC**|:

```
$!GETAUXDATA |ABC|
   AUXDATALOCATION = zone
   NAME = 'ABC.Aux.Data'
   ZONE  = 2
```

$!GETCONNECTIVITYREFCOUNT

Syntax: **$!GETCONNECTIVITYREFCOUNT** *<macrovar>*
    ZONE = *<integer>*
    [no optional parameters]

Description: Fetch the count of how many zones share connectivity with the specified zone. Count includes specified zone.

Required Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	

Example: Fetch the connectivity count from Zone 2, and store it in the macro variable |**ABC**|. If zones 2, 5 and 6 share connectivity, |**ABC**| **= 3**.:

```
$!GETCONNECTIVITYREFCOUNT |ABC|
   ZONE  = 2
```

$!GETCURFRAMENAME

Syntax:

```
$!GETCURFRAMENAME  <macrovar>
   [no parameters]
```

Description: Query Tecplot for the name of the current frame. The *<macrovar>* represents the macro variable to receive the results.

Example: Put the name of the current frame into the macro variable |**CFRAME**|**.**

```
$!GETCURFRAMENAME |CFRAME|
```

$!GETFIELDVALUE

Syntax:

```
$!GETFIELDVALUE <macrovar>
   ZONE  = <integer>
   VAR   = <integer>
   INDEX   = <integer>
```

Description: Fetch the field value (data set value) at the specified point index and assign the value to *<macrovar>*. If the zone referenced is IJ- or IJK-ordered, then the point index is calculated by treating the 2- or 3-dimensional array as a 1-D array.

Required Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	
VAR	= *<integer>*	
INDEX	= *<integer>*	

Example: A data set contains 2 zones and 3 variables. Zone 2 is dimensioned 5 by 3. Fetch the value from variable 3 at I-, J-location 2, 2, and store it in the macro variable |**ABC**|:

```
$!GETFIELDVALUE |ABC|
   ZONE  = 2
   VAR   = 3
   INDEX = 7
```

Note: **INDEX** was calculated using:

```
INDEX = I + (J-1)*|MAXI| + (K-1) * |MAXI| * |MAXJ|
                 = 5*(2-1)+2
                  = 7
```

$!GETFIELDVALUEREFCOUNT

Syntax:

```
$!GETFIELDVALUEREFCOUNT <macrovar>
   ZONE  = <integer>
   VAR   = <integer>
   [no optional parameters]
```

Description: Get the count of how zones many share the indicated variable with the specified zone. Count includes the specified zone.

Required Parameters:

Parameter Syntax		Notes
ZONE	= *<integer>*	
VAR	= *<integer>*	

Example: A data set contains 5 zones and 3 variables. Zones 1, 2 and 4 share variable 3, and zones 3 and 5 share variable three.

```
$!GETFIELDVALUEREFCOUNT |ABC|
   ZONE   = 2
   VAR    = 3
```

This returns |ABC| = 3, while

```
$!GETFIELDVALUEREFCOUNT |DEF|
   ZONE   = 5
   VAR    = 3
```

returns |DEF| = 2 because the variable is not shared across all five zones.

$!GETNODEINDEX

Syntax:

```
$!GETNODEINDEX = <macrovar>
   ZONE = <integer>
   ELEMENT = <integer>
   CORNER = <integer>
   [no optional parameters]
```

Description: This function only works for finite-element zones. Query for the node index in the specified location as described by the **ZONE, ELEMENT,** and **CORNER** parameters.

Required Parameter:

Parameter Syntax	Notes
ZONE = *<integer>*	Zone must be greater than or equal to one.
ELEMENT = *<integer>*	Must be greater than or equal to one and less than or equal to |MAXJ|.
CORNER = *<integer>*	Possible values are 1-3, 1-4, or 1-8, depending upon the element type.

Example: Get the index for the node at corner 3 of the last element in zone number 1.

```
$!GETZONETYPE |ZONETYPE|
   ZONE = 1
$!IF "|ZONETYPE|" = "ORDERED"
   $!GETNODEINDEX |INDEX|
      ZONE = 1
      ELEMENT = |MAXJ|
   CORNER = 3
   ... Do something with |INDEX|...
$!ENDIF
```

$!GETVARLOCATION

Syntax:

```
$!GETVARLOCATION <macrovar>
    ZONE = <integer>
    VAR   = <integer>
```

Description: Returns the location of the variable in the zone as either CELLCENTERED or NODAL and saves in the macro variable.

Required Parameter:

Parameter Syntax	Notes
ZONE = *<integer>*	
VAR = *<integer>*	

Example: Get the variable location for the variable three in zone 1.

$!GETVARNLOCATION |ABC|

```
ZONE = 3
VAR = 1
```

$!GETVARNUMBYNAME

Syntax:

```
$!GETVARNUMBYNAME <macrovar>
   NAME = <string>
```

Description: Given a variable name, get the number for that variable. This variable number can then be used to assign attributes, such as what variable to use for contouring.

Required Parameter:

Parameter Syntax	Notes
NAME = *<string>*	Name of the variable. If a variable has aliases, the name must correspond to one of the aliases.

Example: Get the variable number for the variable named **PRESSURE** and make it the contouring variable.

```
$!GETVARNUMBYNAME |PVARNUM|
  NAME = "PRESSURE"
$!GLOBALCONTOUR
  VAR = |PVARNUM|
```

$!GETZONETYPE

Syntax:

```
$!GETZONETYPE = <macrovar>
   ZONE = <integer>
   [no optional parameters]
```

Description: Query for the zone type of the specified zone. The zone type will be assigned to *<macrovar>*. The possible return values are:

```
"ORDERED"
"FETRIANGLE"
"FEQUAD"
"FETETRA"
"FEBRICK"
```

Required Parameter:

Parameter Syntax	Notes
`ZONE = <integer>`	Zone must be greater than or equal to one.

Example:

```
$!GETZONETYPE |ZONETYPE|
   ZONE = 1
$!IF "|ZONETYPE|" == "FETRIANGLE"
   $!PAUSE "The zone is FE-Triangle."
$!ENDIF
```

$!GLOBALCONTOUR

Syntax:

```
$!GLOBALCONTOUR [<contourgroup>]
   [optional parameters]
```

Description: A SetValue command that changes global attributes associated with contour plots or contour levels. <contourgroup> refers to the defined contour groups, C1-C4, allowed in Tecplot, and takes an integer value of one through four. The <contourgroup> parameter is optional, and if omitted, C1 will be treated as current.

The **NUMBERFORMAT** setting for **LABELS** also controls the number format in the legend.

Optional Parameters:

Parameter Syntax		Notes
VAR	= *<integer>*	Variable used for contour levels.
LABELS		
{		
SHOW	= *<boolean>*	
GENERATEAUTOLABELS	= *<boolean>*	If **TRUE**, automatic labels are repositioned on each redraw.
ALIGNAUTOLABELS	= *<boolean>*	If **TRUE**, automatic labels are aligned with the contour lines, otherwise they are horizontal.
LABELWITHVALUE	= *<boolean>*	If **TRUE**, automatic labels show the contour value otherwise they show the contour level number.
AUTOLEVELSKIP	*<op>* *<integer>*	Value is in Y-frame units.
AUTOLABELSPACING	*<op>* *<dexp>*	
COLOR	= *<color>*	
ISFILLED	= *<boolean>*	
FILLCOLOR	= *<color>*	
MARGIN	*<op>* *<dexp>*	
TEXTSHAPE	*<<textshape>>*	Not allowed to change size units parameter.
NUMFORMAT	*<<numberformat>>*	
}		
LEGEND		
{		
LABELLOCATION	= *<<contlabellocation>>*	
LABELINCREMENT	= *<double>*	
ANCHORALIGNMENT	= *<anchoralignment>*	
SHOW	= *<boolean>*	
SHOWHEADER	= *<boolean>*	
ROWSPACING	*<op>* *<dexp>*	
ISVERTICAL	= *<boolean>*	
OVERLAYBARGRID	= *<boolean>*	
TEXTCOLOR	= *<color>*	Thin line around each band in the color bar.
XYPOS	*<<xy>>*	
BOX	*<<textbox>>*	
HEADERTEXTSHAPE	*<<textshape>>*	
NUMBERTEXTSHAPE	*<<textshape>>*	
AUTORESIZE	= *<boolean>*	
AUTOSIZEMAXLIMIT	= *<double>*	
CONTCOLORLABELDELTA	= *<double>*	
INCLUDECUTOFFLEVELS	= *<boolean>*	Set only via config file.
}		
COLORCUTOFF		Set minimum and maximum cutoff for contour flooding.
{		
RANGEMIN	*<op> <dexp>*	
RANGEMAX	*<op> <dexp>*	
INCLUDEMIN	*= <boolean>*	
INCLUDEMAX	*= <boolean>*	
}		

Parameter Syntax	Notes
CONTOURLINESTYLE **{** **CONTOURLINEMODE** = *<contourlinemode>* **LINESKIP** *<op> <integer>* **PATTERNLENGTH** *<op> <dexp>* **}**	This is used to assign a special line pattern scheme for contour line plots.
COLORMAPFILTER **{** **REVERSECOLORMAP** = *<boolean>* **COLORMAPCYCLES** *<op> <integer>* **COLORMAPOVERRIDEACTIVE** = *<boolean>* **COLORMAPOVERRIDE** *<integer>* *<<colormapoverride>>* **ZEBRA** *<<zebrashade>>* **COLORMAPDISTRIBUTION** *<colormapdistribution>* **CONTINUOUSCOLOR** *<<continuouscolor>>* **USEFASTSPPROXCONTINUOUSFLOOD** = *<boolean>* **}**	The global color map is defined using the **$!COLORMAP** command. **COLORMAPFILTER** allows each frame to make adjustments to the global color map that will only apply to the current frame. Use *<integer>* to choose which override to operate on. Default = FALSE
DEFNUMLEVELS = *<integer>*	Sets the target number of contour levels for situations where contour levels are automatically reset. Tecplot will attempt to create levels where the start, end and increment values are all clipped floating point values.

Example: This example does the following: Turns on the contour legend; Sets the flood cutoff to go from 3 to 5; Reverses the color map; Inserts a color map override of yellow between contour level number 7 and level number 9.

```
$!GLOBALCONTOUR [1]
   LEGEND
   {
    SHOW = YES
   }
   COLORCUTOFF
   {
    RANGEMIN  = 3
    RANGEMAX  = 5
    INCLUDEMIN   = TRUE
    INCLUDEMAX = TRUE
   }
   COLORMAPFILTER
   {
    REVERSECOLORMAP = TRUE
    COLORMAPOVERRIDEACTIVE = TRUE
    COLORMAPOVERRIDE 1
```

```
 {
  INCLUDE    = YES
  COLOR      = YELLOW
  STARTLEVEL = 7
  ENDLEVEL   = 9
 }
}
```

$!GLOBALFRAME

Syntax: `$!GLOBALFRAME`
[optional parameters]

Description: A SetValue command that sets attributes which apply to all frames.

Optional Parameters:

Parameter Syntax		Notes
FRAMEHEADERHEIGHT	*<op> <dexp>*	Value is in inches.
SNAPTOGRID	= *<boolean>*	Even if set to **TRUE**, Tecplot may not allow snapping in some situations.
FRAMEHEADERFORMAT	= *<string>*	The *<string>* contains the text that appears in each of Tecplot's frame headers. This string typically contains dynamic text. See the Tecplot User's Manual. The default string is: **"&(FRAMENAME)|&(DATE)|&(DATASETTITLE)."**
SNAPTOPAPER	= *<boolean>*	Even if set to **TRUE**, Tecplot may not allow snapping in some situations.

Example: Customize the frame header text, and set the frame header height to be 0.25 inches:

```
$!GLOBALFRAME
   FRAMEHEADERFORMAT = "My frame, the current date is
   &(Date), &(Time)"
   FRAMEHEADERHEIGHT = 0.25
```

$!GLOBALISOSURFACE

Syntax: `$!GLOBALISOSURFACE`

[optional parameters]

Description: A SetValue command which changes global attributes associated with iso-surfaces.

Optional Parameters:

Parameter Syntax	Notes
SHOW = *<boolean>*	
ISOSURFACESELECTION = *<isosurfaceselection>*	
ISOVALUE1 = *<double>*	
ISOVALUE2 = *<double>*	
ISOVALUE3 = *<double>*	

MESH
{
SHOW = *<boolean>*
COLOR = *<color>*
LINETHICKNESS = *<double>*
}	

CONTOUR
{
SHOW = *<boolean>*
USELIGHTINGEFFECT = *<boolean>*
CONTOURTYPE = *<contourtype>*
FLOODCOLORING = *<contourcoloring>*
LINECONTOURGROUP = *<sminteger>*
COLOR = *<color>*
LINETHICKNESS = *<double>*
}
	Default = **FLOOD**, **PRIMARYVALUE** and **AVERAGECELL** not allowed.
	Default = **Group1**

SHADE
{
SHOW = *<boolean>*
COLOR = *<color>*
USELIGHTINGEFFECT = *<boolean>*
}	

SURFACEEFFECTS
{
LIGHTINGEFFECT = *<lightingeffect>*
SURFACETRANSLUCENCY = *<translucency>*
USETRANSLUCENCY = *<boolean>*
}	

DEFINITIONCONTOURGROUP = *<sminteger>*	Contour group from which iso-surfaces are based. Default = 1
MARCHINGCUBEALGORITHM = *[classic or classicplus]*	

$Example:

```
!GLOBALISOSURFACE
   ISOSURFACESELECTION = ONESPECIFICVALUE
   ISOVALUE1 = 113.626812744
   MESH{SHOW = YES}
   MESH{COLOR = BLUE}
   MESH{LINETHICKNESS = 0.4}
   CONTOUR{SHOW = YES}
   SURFACEEFFECTS{LIGHTINGEFFECT = PANELED}
   SURFACEEFFECTS{SURFACETRANSLUCENCY = 60}
```

$!GLOBALLINEPLOT

Syntax: `$!GLOBALLINEPLOT`
  [optional parameters]

Description: A SetValue command that changes global attributes associated with Line-plots.

Optional Parameters:

Parameter Syntax		Notes
DATALABELS **{**		These are text values that can be added to a plot to show the indices or values for the data points.
SHOWNODELABELS	= *<boolean>*	
COLOR	= *<color>*	
INCLUDEBOX	= *<boolean>*	
NODELABELTYPE	= *<nodelabeltype>*	
INDEXSKIP	*<op>* *<integer>*	
DISTANCESKIP	*<op>* *<dexp>*	
SKIPMODE	= *<skipmode>*	
TEXTSHAPE	*<<textshape>>*	
NUMFORMAT	*<<numberformat>>*	Not allowed to change size units parameter.
COLORBYZONEMAP	= *<boolean>*	
}		
LEGEND **{**		Attributes for an optional legend added to an Line-plot. Entries in the legend are determined dynamically by Tecplot depending on which mappings are turned on.
SHOW	= *<boolean>*	
SHOWTEXT	= *<boolean>*	
TEXTCOLOR	= *<color>*	
ROWSPACING	*<op>* *<dexp>*	
TEXTSHAPE	*<<textshape>>*	Not allowed to change size units.
BOX	*<<textbox>>*	
XYPOS	*<<xy>>*	
ANCHORALIGNMENT	= *<anchoralignment>*	
}		

Example: Turn on the data labels and show the Line-legend. Use the **TIMESBOLD** font in the legend:

```
$!GLOBALLINEPLOT
   DATALABELS
   {
    SHOWNODELABELS = YES
   }
   LEGEND
   {
    SHOW = YES
    TEXTSHAPE
    {
     FONT = TIMESBOLD
    }
   }
```

$!GLOBALPOLAR

Syntax: `$!GLOBALPOLAR`

[optional parameters]

Description: Allows polar plots to have curved lines that are interpolated along the R-Axis between data points.

Optional Parameters:

Parameter Syntax		Notes
DRAWSTRAIGHTLINES	= <*boolean*>	Default=TRUE. Alternates between straight and curved interpolated lines for polar plots.
ANGLE	= <*float*>	Default=1.0. Determines the angle for which lines will be approximated as curves.

Example: This example turns on curved lines and defines the maximum angle to be approximated as a curved line to be 2.0 degrees..

```
$!GLOBALPOLAR
   DRAWSTRAIGHTLINES = FALSE
   ANGLE = 2.0
```

Example: **$!GLOBALRGB**

Syntax:

```
$!GLOBALRGB
   RGBMode = <RGBMode>
```

[optional parameters]

Description: Allows RGB coloring for plots which have RGB values specified at each vertex. This coloring option is valuable for plots with entities such as Gas, Oil and Water. RGB Coloring can be assigned to field plot objects such as zones, iso-surfaces and slices

Required Parameter:

Parameter Syntax		Notes

`RGBMODE`	= `SpecifyRGB`
	`SpecifyRG`
	`SpecifyRB`
	`SpecifyGB`
		Sets whether the user specifies all three color variables for RGB Coloring, or if Tecplot calculates one variable while the user specifies two.

Optional Parameters:

Parameter Syntax		Notes
`REDCHANNELVAR`	= <*integer*>	Sets variable for the red channel.
`GREENCHANNELVAR`	= <*integer*>	Sets variable for the green channel.
`BLUECHANNELVAR`	= <*integer*>	Sets variable for the blue channel.
`RANGEMIN`	= <*double*>	Default=0.0

Parameter Syntax		Notes
RANGEMAX	= *<double>*	Default=1.0

LEGEND
{
SHOW
SHOWLABELS
TEXTCOLOR
HEIGHT
XYPOS
TEXTSHAPE
BOX
AHCHOR
USEREDVARNAME
REDCHANNELLABEL
USEGREENVARNAME
GREENCHANNELLABEL
USEBLUEVARNAME
BLUECHANNELLABEL
RGBLEGENDORIENTATION
}
	= *<boolean>*
	= *<boolean>*
	= *<color>*
	= *<double>*
	= *<<xy>>*
	= *<<textshape>>*
	= *<<textbox>>*
	= *<anchoralighnment>*
	= *<boolean>*
	= *<string>*
	= *<boolean>*
	= *<string>*
	= *<boolean>*
	= *<string>*
	= **[OrientRGB,
	OrientGBR,
	OrientBRG,
	OrientRBG,
	OrientBGR,
	OrientGRB]**	

Example: This example turns on RGB Coloring and defines variables for the Red and Green Channel, leaving Tecplot to calculate the Blue Channel values.

```
$!GLOBALRGB
   RGBMODE = SPECIFYRG
   REDCHANNELVAR = 1
   GREENCHANNELVAR = 4
```

$!GLOBALSCATTER

Syntax: **$!GLOBALSCATTER**
[optional parameters]

Description: A SetValue command that changes global attributes associated with scatter plots.

Optional Parameters:

Parameter Syntax		Notes
VAR	= *<integer>*	Scatter sizing variable.
RELATIVESIZE	*<op>* *<dexp>*	Scaling factor for scatter symbols sized "By Variable."

Parameter Syntax	Notes
RELATIVESIZEINGRIDUNITS = <boolean>	If **TRUE**, scatter sizing “By Variable” is in grid units / magnitude otherwise centimeters/magnitude.
BASEFONT = *<font>*	
LEGEND **{** **SHOW** = *<boolean>* **SHOWTEXT** = *<boolean>* **TEXTCOLOR** = *<color>* **ROWSPACING** *<op> <dexp>* **TEXTSHAPE** *<<textshape>>* **BOX** *<<textboxtype>>* **ANCHORPOS** *<<anchorpos>>* **}**	Not allowed to change size units parameter.
REFSCATSYMBOL **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **ISFILLED** = *<boolean>* **FILLCOLOR** = *<color>* **LINETHICKNESS** *<op> <dexp>* **MAGNITUDE** *<op> <dexp>* **XYPOS** *<<xy>>* **SYMBOLSHAPE** *<<symbolshape>>* **}**	
DATALABELS **{** **SHOWNODELABELS** = *<boolean>* **SHOWCELLLABELS** = *<boolean>* **COLOR** = *<color>* **INCLUDEBOX** = *<boolean>* **NODELABELTYPE** = *<nodelabeltype>* **NODELABELVAR** *<op> <integer>* **INDEXSKIP** *<op> <integer>* **DISTANCESKIP** *<op> <dexp>* **SKIPMODE** = *<skipmode>* **TEXTSHAPE** *<<textshape>>* **NUMFORMAT** *<<numberformat>>* **CELLLABELTYPE** = *<labeltype_e>* **CELLLABELVAR** = *<entindex_t>* **COLORBYZONEMAP** = *<boolean>* **}**	These are text labels that can be added to a plot to show node or cell values. Not allowed to change size units parameter.
SPHERESCATTERRENDER QUALITY = *<spherescatterrender quality>*	Takes values **LOW**, **MEDIUM**, or **HIGH**. Config file only option.

Example: This example does the following:

- Increases the relative size of scatter symbols that are sized by variable by ten percent.
- Turns on the scatter sizing legend.

- Turns on the reference scatter symbol and makes it red.
- Turns on data labels for nodes.

```
$!GLOBALSCATTER
   RELATIVESIZE * = 1.1
   LEGEND
   {
    SHOW = YES
   }
   REFSCATSYMBOL
   {
    SHOW  = YES
    COLOR = RED
   }
   DATALABELS
   {
    SHOWNODELABELS = TRUE
   }
```

$!GLOBALSLICE

Syntax: `$!GLOBALSLICE`
  [optional parameters]

Description: A SetValue command that changes global attributes associated with streamtraces.

Optional Parameters:

Parameter Syntax		Notes
`SHOW`	= *<boolean>*	
`SHOWPOSITION2`	= *<boolean>*	
`SHOWINTERMEDIATESLICES`	= *<boolean>*	
`NUMINTERMEDIATESLICES`	= *<integer>*	
`SLICESURFACE`	= *<slicesurface>*	

`POSITION1`
`{`
` X`
` Y`
` Z`
` I`
` J`
` K`
`}`
	
	= *<double>*
	= *<double>*
	= *<double>*
	= *<integer>*
	= *<integer>*
	= *<integer>*	

Parameter Syntax	Notes
POSITION2 **{** **X** = *<double>* **Y** = *<double>* **Z** = *<double>* **I** = *<integer>* **J** = *<integer>* **K** = *<integer>* **}**	
MESH **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **LINETHICKNESS** = *<double>* **}**	

CONTOUR
{
SHOW = *<boolean>*
CONTOURTYPE = *<contourplottype>*
COLOR = *<color>*
LINETHICKNESS = *<double>*
USELIGHTINGEFFECT = *<boolean>*
FLOODCOLORING = *<contourcoloring_e>*
LINECONTOURGROUP = *<sminteger_t>*
}
	CORNERCELL and **AVERAGECELL** options not allowed for **CONTOURTYPE**.
	Default = **Group1**
	Default = **1**

SHADE **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **USELIGHTINGEFFECT** = *<boolean>* **}**	
VECTOR **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **ISTANGENT** = *<boolean>* **LINETHICKNESS** = *<double>* **VECTORTYPE** = *<vectorplottype>* **ARROWHEADSTYLE** = *<arrowheadstyle>* **}**	
BOUNDARY **{** **SHOW** = *<boolean>* **COLOR** = *<color>* **LINETHICKNESS** = *<op><dexp>* **}**	
SURFACEEFFECTS **{** **LIGHTINGEFFECT** = *<lightingeffect>* **SURFACETRANSLUCENCY** = *<translucency>* **USETRANSLUCENCY** = *<boolean>* **}**	

Example:

```
$!GLOBALSLICE POSITION1 {X = 6}
$!GLOBALCONTOUR VAR = 4
$!GLOBALSLICE SHOW = YES
$!GLOBALSLICE POSITION2 {X = 1}
$!GLOBALSLICE SHOWPOSITION2 = YES
$!GLOBALSLICE SHOWINTERMEDIATESLICES = YES
$!GLOBALSLICE NUMINTERMEDIATESLICES = 6
$!REDRAW
$!CREATESLICEZONES
```

$!GLOBALSTREAM

Syntax: `$!GLOBALSTREAM`
[optional parameters]

Description: A SetValue command that changes global attributes associated with streamtraces.

Optional Parameters:

Parameter Syntax		Notes
`SHOW`	`=` *<boolean>*	
`ADDARROWS`	`=` *<boolean>*	
`CELLFRACTION`	*<op> <dexp>*	Maximum fraction of the distance across a cell a streamtrace moves in one step. A streamtrace adjusts its step-size between `CELLFRACTION` and `MINCELLFRACTION` depending on local curvature of the streamtrace.
`MINCELLFRACTION`	*<op> <dexp>*	Minimum fraction of the distance across a cell a streamtrace moves in one step.
`ARROWHEADSIZE`	*<op> <dexp>*	
`ARROWHEADSPACING`	*<op> <double>*	Distance between arrowheads in frame units.

Parameter Syntax		Notes

RODRIBBON
{
WIDTH
NUMRODPOINTS
MESH
{
SHOW
COLOR
LINETHICKNESS
}
CONTOUR
{
SHOW
USELGHTINGEFFECT
FLOODCOLORING
}
SHADE
{
SHOW
COLOR
USELIGHTINGEFFECT
}
SURFACEEFFECT
{
LIGHTINGEFFECT
SURFACETRANSLUCENCY
USETRANSLUCENCY
}
}
	<op><dexp>
	<op> <integer>
	= <boolean>
	= <color>
	<op><dexp>
	= <boolean>
	= <boolean>
	= <contourcoloring_e>
	= <boolean>
	= <color>
	= <boolean>
	= <lightingeffect>
	= <translucency>
	= <boolean>
		Value is grid units.
		Number of points used to define the streamrod cross-section.

LINETHICKNESS	*<op> <dexp>*	
MAXSTEPS	*<op> <integer>*	
COLOR	*= <color>*	

Parameter Syntax		Notes

STREAMTIMING
{
DOTIMEMARKS
DOTIMEDASHES
DELTATIME
STARTTIME
ENDTIME
MARKCOLOR
MARKSIZE
DASHSKIP
MARKSYMBOL
}
	
	= *<boolean>*
	= *<boolean>*
	<op> *<dexp>*
	<op> *<dexp>*
	<op> *<dexp>*
	= *<color>*
	<op> *<dexp>*
	<op> *<integer>*
	<<symbolshape>>	

TERMLINE
{
ISACTIVE
SHOW
COLOR
LINEPATTERN
PATTERNLENGTH
LINETHICKNESS
}
	
	= *<boolean>*
	= *<boolean>*
	= *<color>*
	= *<linepattern>*
	<op> *<dexp>*
	<op> *<dexp>*
		Use the **$!STREAMTRACE** action command to define the stream termination polyline.

$!GLOBALTHREED

Syntax: **$!GLOBALTHREED**
[optional parameters]

Description: A SetValue command that changes global attributes associated with 3-D plots.

Optional Parameters:

Parameter Syntax		Notes
PERFORMEXTRA3DSORTING	*<boolean>*	
AXISBOXPADDING	*<op>* *<dexp>*	
LINELIFTFRACTION	*<op>* *<dexp>*	
SYMBOLLIFTFRACTION	*<op>* *<dexp>*	
VECTORLIFTFRACTION	*<op>* *<dexp>*	

SLICE
{
ORIGIN
NORMAL
}
	
	<<xyz>>
	<<xyz>>	

Parameter Syntax		Notes
AXISSCALEFACT	*<<xyz>>*	The 3-D axis must be **INDEPENDENT** for this option to work properly. See **$!THREEDAXIS**.
ROTATEORIGIN	*<<xyz>>*	
LIGHTSOURCE **{**		Always specify all three components here. Tecplot normalizes X, Y and Z after processing the Z-component. X, Y and Z represent a vector in the eye coordinate system.
XYZDIRECTION	*<<xyz>>*	
INTENSITY	= *<double>*	
BACKGROUNDLIGHT	= *<double>*	
SURFACECOLORCONTRAST	= *<double>*	
INCLUDESPECULAR	= *<boolean>*	Default = **FALSE**
SPECULARINTENSITY	= *<integer>*	Range = 1-100
SPECULARSHININESS	= *<integer>*	Range = 1-100
}		
FORCEGOURADFOR3DCONTFLOOD	= *<boolean>*	Default = **TRUE**
FORCEPANELEDFOR3DCELLFLOOD	= *<boolean>*	Default = **TRUE**

Example:

```
$!GLOBALTHREED ROTATEORIGIN{X = 4.36052333891}
   $!GLOBALTHREED
   LIGHTSOURCE
    {
      XYZDIRECTION
       {
         X = 0.398226616447
         Y = 0.435028248588
         Z = 0.807567944438
       }
   $!GLOBALTHREED LIGHTSOURCE{INTENSITY = 80}
   $!GLOBALTHREED LIGHTSOURCE{BACKGROUNDLIGHT = 25}
   $!GLOBALTHREED LIGHTSOURCE{SURFACECOLORCONTRAST =
   85}
   $!GLOBALTHREED LINELIFTFRACTION = 7
   $!GLOBALTHREED SYMBOLLIFTFRACTION = 0.5
   $!GLOBALTHREED VECTORLIFTFRACTION = 6
   $!GLOBALTHREED PERFORMEXTRA3DSORTING = YES
```

$!GLOBALTHREEDVECTOR

Syntax: **$!GLOBALTHREEDVECTOR**
[optional parameters]

Description: A SetValue command that changes global attributes associated with 3-D vector plots.

Optional Parameters:

Parameter Syntax		Notes
RELATIVELENGTH	*<op> <dexp>*	
UNIFORMLENGTH	*<op> <dexp>*	Value is in Y-frame units.
USERELATIVE	= *<boolean>*	If **FALSE**, vectors are all the same size (**UNIFORMLENGTH**).
RELATIVELENGTHINGRIDUNITS	= *<boolean>*	If **TRUE** and **USERELATIVE** is **TRUE** then vectors are sized in Grid Units/Magnitude. If **FALSE** and **USERELATIVE** is **TRUE** then vectors are sized in cm/magnitude.
HEADSIZEASFRACTION	*<op> <dexp>*	Head is sized as a fraction of the stem length.
HEADSIZEINFRAMEUNITS	*<op> <dexp>*	Value is in Y-frame units.
SIZEHEADBYFRACTION	= *<boolean>*	If **TRUE**, **HEADSIZEASFRACTION** is used to size arrowheads otherwise **HEADSIZEINFRAMEUNITS** is used.
ARROWHEADANGLE	*<op> <dexp>*	Angle is in degrees.
UVAR	= *<integer>*	Variable number for the X-vector component.
VVAR	= *<integer>*	Variable number for the Y-vector component.
WVAR	= *<integer>*	Variable number for the Z-vector component.
REFVECTOR **{** **SHOW** **COLOR** **MAGNITUDE** **LINETHICKNESS** **ANGLE** **XYPOS** **MAGNITUDELABEL** **{** **SHOW** **TEXTCOLOR** **TEXTSHAPE** **NUMFORMAT** **OFFSET** **}**	 = *<boolean>* = *<color>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<<xy>>* = *<boolean>* = *<color>* *<<textshape>>* *<<numberformat>>* = *<double>*	

Example: This example does the following:

- Makes all vectors be uniform in size; 5 percent in Y-frame units.
- Makes the arrowheads 0.2 times the size of the stems.
- Turns off the reference vector.

```
$!GLOBALTHREEDVECTOR
   USERELATIVE = FALSE
```

```
UNIFORMLENGTH = 5
HEADSIZEASFRACTION = .2
REFVECTOR
{
 SHOW = FALSE
}
```

$!GLOBALTWODVECTOR

Syntax: **$!GLOBALTWODVECTOR**
[optional parameters]

Description: A SetValue command that changes global attributes associated with 2-D vector plots.

Optional Parameters:

Parameter Syntax	Notes
RELATIVELENGTH *<op> <dexp>*	
UNIFORMLENGTH *<op> <dexp>*	Value is in Y-frame units.
USERELATIVE = *<boolean>*	If **FALSE**, vectors are all the same size (**UNIFORMLENGTH**).
RELATIVELENGTHINGRIDUNITS = *<boolean>*	If **TRUE** and **USERELATIVE** is **TRUE** then vectors are sized in Grid Units/Magnitude. If **FALSE** and **USERELATIVE** is **TRUE** then vectors are sized in centimeters/magnitude.
HEADSIZEASFRACTION *<op> <dexp>*	Head is sized as a fraction of stem length.
HEADSIZEINFRAMEUNITS *<op> <dexp>*	Value is in Y-frame units.
SIZEHEADBYFRACTION = *<boolean>*	If **TRUE**, **HEADSIZEASFRACTION** is used to size arrowheads other **HEADSIZEINFRAMEUNITS** is used.
ARROWHEADANGLE *<op> <dexp>*	Angle is in degrees.
UVAR *<op> <integer>*	Variable number for the X-vector component.

Parameter Syntax		Notes
VVAR	*<op> <integer>*	Variable number for the Y-vector component.
REFVECTOR **{** **SHOW** **COLOR** **MAGNITUDE** **LINETHICKNESS** **ANGLE** **XYPOS** **MAGNITUDELABEL** **{** **SHOW** **TEXTCOLOR** **TEXTSHAPE** **NUMFORMAT** **OFFSET** **}**	 *= <boolean>* *= <color>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<<xy>>* *= <boolean>* *= <color>* *<<textshape>>* *<<numberformat>>* *= <double>*	

Example: This example does the following:

- Doubles the vector length (assume vectors currently drawn using relative length).
- Make the vector heads uniform in size; 2 percent in frame units.
- Make the head angle 15 degrees.

```
$!GLOBALTWODVECTOR
   RELATIVELENGTH      * = 2
   SIZEHEADBYFRACTION    = NO
   HEADSIZEINFRAMEUNITS = 2
   HEADANGLE             = 15
```

$!IF...$!ENDIF

Syntax: **$!IF** *<conditionalexp>*
$!ENDIF

Description: Conditionally process macro commands.

Example 1: Process macro commands if the macro variable |**myvar**| is less than 73.2:

```
$!IF |myvar| < 73.2
   .
   .
   .
$!ENDIF
```

Example 2: Process macro commands if the macro variable |**response**| is **YES**:

```
$!IF "|response|" == "YES"
   .
   .
   .
$!ENDIF
```

$!INCLUDEMACRO

Syntax: `$!INCLUDEMACRO` *<string>*

Description: Insert the commands from another macro file. Because the **`$!INCLUDEMACRO`** command is processed when the macro is loaded and not when the macro is executed, you are not allowed to reference macro variables within the ***<string>*** parameter.

Example: Include the macro file **`m2.mcr`**:

```
$!INCLUDEMACRO "m2.mcr"
```

$!INTERFACE

Syntax: `$!INTERFACE`
   [optional parameters]

Description: A SetValue command that sets attributes related to the Tecplot interface.

Optional Parameters:

Parameter Syntax	Notes
ALLOWDATAPOINTSELECT = *<boolean>*	If TRUE, Tecplot allows you to use the Adjustor tool to select and move data points.
APPROXIMATIONMODE = *<boolean>*	If TRUE, Tecplot allows you to use the Adjustor tool to select and move data points.
AUTOREDRAWISACTIVE= *<boolean>*	Set to FALSE to turn Auto Redraw off.
BACKINGSTOREMODE = *<backingstoremode>*	
BEEPONFRAMEINTERRUPT = *<boolean>*	

Parameter Syntax		Notes
CACHELIGHTDISPLAYLISTSONLY = *<boolean>*		When caching graphics in display lists, only cache those objects which uses little memory. When this is on, only approximated plots are saved. Full plots are not saved. This only has an effect if USEDISPLAYLISTS is set to TRUE, and if USEAPPROXIMATEPLOTS is TRUE.
CONSERVEDERIVEDVARIABLESPACE	= *<boolean>*	
DATA **{** **SMOOTHBNDRYCOND** **NUMSMOOTHPASSES** **SMOOTHWEIGHT** **INVDISTEXPONENT** **INVDISTMINRADIUS** **LINEARINTERPCONST** **LINEARINTERPMODE** **INTERPPTSELECTION** **INTERPNPOINTS** **KRIGRANGE** **KRIGZEROVALUE** **KRIGDRIFT** **DERIVATIVEBOUNDARY** **TRIANGLEKEEPFACTOR** **VARIABLEDERIVATIONMETHOD** **CONTLINECREATEMODE** **}**	= *<boundarycondition>* *<op> <integer>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* = *<linearinterpmode>* = *<pointselection>* *<op> <integer>* *<op> <dexp>* *<op> <dexp>* = *<drift>* = *<derivpos>* *<op> <dexp>* = *<ACCURATE or FAST>* = *<ONEZONEPERCONTOURLEVER or ONEZONEPERINDEPENDENTPOLYLINE>*	Settings for smoothing and interpolation. Default = ACCURATE Note that this is a config file option only.
ENABLEDELAYS	= *<boolean>*	Enable or disable delays in macro commands.
ENABLEINTERRUPTS	= *<boolean>*	Enable or disable user interrupts.
ENABLEPAUSES	= *<boolean>*	Enable or disable pause.
ENABLEWARNINGS	= *<boolean>*	Enable or disable warning dialogs.
FEBOUNDARYUSESVALUEBLANKING	= *<boolean>*	

Parameter Syntax	Notes
DIALOGPLACEMENT { ADVANCED3DCONTROLDIALOG <<*dialogplacement*>> AXISEDITDIALOG <<*dialogplacement*>> COLORMAPDIALOG <<*dialogplacement*>> CONTOURDIALOG <<*dialogplacement*>> CREATE1DLINEDIALOG <<*dialogplacement*>> CREATECIRCULARZONEDIALOG <<*dialogplacement*>> CREATERECTANGULARZONEDIALOG <<*dialogplacement*>> CREATEZONEFROM POLYLINESDIALOG <<*dialogplacement*>> CREATEZONEFROMVALUESDIALOG <<*dialogplacement*>> CURVEINFODIALOG <<*dialogplacement*>> DATAINFODIALOG <<*dialogplacement*>> DATALABELSDIALOG <<*dialogplacement*>> DATASPREADSHEETDIALOG <<*dialogplacement*>> DELETEVARIABLESDIALOG <<*dialogplacement*>> DELETEZONESDIALOG <<*dialogplacement*>> DEPTHBLANKINGDIALOG <<*dialogplacement*>> DUPLICATEZONEDIALOG <<*dialogplacement*>> EQUATIONDIALOG <<*dialogplacement*>> EXPORTDIALOG <<*dialogplacement*>> EXTRACTCONTOURLINESDIALOG <<*dialogplacement*>> EXTRACTDISCRETEPOINTSDIALOG <<*dialogplacement*>> EXTRACTFEBOUNDARYDIALOG <<*dialogplacement*>> EXTRACTISOSURFACESDIALOG <<*dialogplacement*>> EXTRACTPOINTSFROMGEOMETRYDIALOG <<*dialogplacement*>> EXTRACTPOINTSFROMPOLYLINEDIALOG <<*dialogplacement*>> EXTRACTSLICEFROMPLANEDIALOG <<*dialogplacement*>> EXTRACTSLICESDIALOG <<*dialogplacement*>> EXTRACTSTREAMTRACESDIALOG <<*dialogplacement*>> EXTRACTSUBZONEDIALOG <<*dialogplacement*>> IJKBLANKINGDIALOG <<*dialogplacement*>> IMPORTDIALOG <<*dialogplacement*>> INVERSEDISTANCEINTERPOLATIONDIALOG <<*dialogplacement*>> ISOSURFACESDIALOG <<*dialogplacement*>> KRIGINGINTERPOLATIONDIALOG <<*dialogplacement*>> LIGHTSOURCEDIALOG <<*dialogplacement*>> LINEARINTERPOLATIONDIALOG <<*dialogplacement*>> LINEMAPLEGENDDIALOG <<*dialogplacement*>> LOADDATADIALOG <<*dialogplacement*>> MACROPLAYDIALOG <<*dialogplacement*>> MACRORECORDDIALOG <<*dialogplacement*>> MACROVIEWERDIALOG <<*dialogplacement*>> MIRRORZONEDIALOG <<*dialogplacement*>> NEWLAYOUTDIALOG <<*dialogplacement*>> OPENLAYOUTDIALOG <<*dialogplacement*>> ORDERFRAMESDIALOG <<*dialogplacement*>> PAPERSETUPDIALOG <<*dialogplacement*>> POLARDRAWINGOPTIONSDIALOG <<*dialogplacement*>> PRINTDIALOG <<*dialogplacement*>> PROBEATDIALOG <<*dialogplacement*>> PROBEDIALOG <<*dialogplacement*>> QUICKEDITDIALOG <<*dialogplacement*>> QUICKMACROPANELDIALOG <<*dialogplacement*>> RESET3DAXESDIALOG <<*dialogplacement*>> RGBCOLORLEGENDDIALOG <<*dialogplacement*>> RGBCOLORVARSANDRANGEDIALOG <<*dialogplacement*>> ROTATE2DDATADIALOG <<*dialogplacement*>>	The DIALOGPLACMENT parameter may only appear in the tecplot config file. You may specify the placement of the indicated dialogs. Dialog placement is relative to the main Tecplot window.

Parameter Syntax	Notes
RULERGRIDDIALOG <<*dialogplacement*>> SAVEASDIALOG <<*dialogplacement*>> SAVEDIALOG <<*dialogplacement*>> SCATTERLEGENDDIALOG <<*dialogplacement*>> SCATTERREFERENCESYMBOLDIALOG <<*dialogplacement*>> SCATTERSIZEANDFONTDIALOG <<*dialogplacement*>> SLICESDIALOG <<*dialogplacement*>> SMOOTHDIALOG <<*dialogplacement*>> SPATIALVARSDIALOG <<*dialogplacement*>> STREAMTRACESDIALOG <<*dialogplacement*>> STYLELINKINGDIALOG <<*dialogplacement*>> THREEDAXISLIMITSDIALOG <<*dialogplacement*>> THREEDORIENTATIONAXISDIALOG <<*dialogplacement*>> THREEDVIEWDETAILSDIALOG <<*dialogplacement*>> THREEDVIEWROTATEDIALOG <<*dialogplacement*>> TRANSFORMCOORDINATESDIALOG <<*dialogplacement*>> TRANSLATEMAGNIFYDIALOG <<*dialogplacement*>> TRIANGULATEDIALOG <<*dialogplacement*>> TWODDRAWORDERDIALOG <<*dialogplacement*>> VALUEBLANKINGDIALOG <<*dialogplacement*>> VECTORARROWHEADSDIALOG <<*dialogplacement*>> VECTORLENGTHDIALOG <<*dialogplacement*>> VECTORREFERENCEVECTORDIALOG <<*dialogplacement*>> VECTORVARSDIALOG <<*dialogplacement*>> WRITEDATADIALOG <<*dialogplacement*>> ZONEMAPSTYLEDIALOG <<*dialogplacement*>>	
INITIALPLOTFIRSTZONEONLY = <*boolean*>	If TRUE, only the first enabled zone is activated. Default shows all zones (except from within a layout).
INITIALPLOTTYPE = <*plottype*>	Default is Automatic
INTERRUPTCHECKINGFREQUENCY = <*integer*>	Set the number of milliseconds between checks for a key- or button-press by the user to interrupt processing in Tecplot.
LISTCOMMANDSINMACROVIEWER = <*boolean*>	If FALSE, macro commands are displayed in full one at a time.
LOADADDONSUSINGLAZYRELOCATE = <*boolean*>	If set to FALSE, all add-on symbols are loaded immediately.
MAXCUSTOMCOLORSININTERFACE = <integer>	UNIX only. Valid values are 1 to 56. Some UNIX displays cannot allocate enough colors for the Tecplot interface. Use this option to limit the number of custom colors displayed in the Tecplot interface.
MAXTRACELINES <*integer*>	Maximum number of lines to use when tracing data in a frame.

Parameter Syntax		Notes
MINPIXELSFORDRAG	*<integer>*	Number of pixels to move the pointer before it is considered a drag.
MOUSEACTIONS		
{		
MIDDLEBUTTON		
{		
BUTTONCLICK	*<mousebuttonclick>*	
SIMPLEDRAG	*<mousebuttondrag>*	
CONTROLLEDDRAG	*<mousebuttondrag>*	
ALTEDDRAG	*<mousebuttondrag>*	
SHIFTEDDRAG	*<mousebuttondrag>*	
CONTROLALTEDDRAG	*<mousebuttondrag>*	
CONTROLSHIFTEDDRAG	*<mousebuttondrag>*	
ALTSHIFTEDDRAG	*<mousebuttondrag>*	
CONTROLALTSHIFTEDDRAG	*<mousebuttondrag>*	
}		
RIGHTBUTTON		
{		
BUTTONCLICK	*<mousebuttonclick>*	
SIMPLEDRAG	*<mousebuttondrag>*	
CONTROLLEDDRAG	*<mousebuttondrag>*	
ALTEDDRAG	*<mousebuttondrag>*	
SHIFTEDDRAG	*<mousebuttondrag>*	
CONTROLALTEDDRAG	*<mousebuttondrag>*	
CONTROLSHIFTEDDRAG	*<mousebuttondrag>*	
ALTSHIFTEDDRAG	*<mousebuttondrag>*	
CONTROLALTSHIFTEDDRAG	*<mousebuttondrag>*	
}		
}		
NUMMOUSEBUTTONS	*<integer>*	This option is only for UNIX users who are using MIDDLEMOUSEBUTTONMODE or RIGHTMOUSEBUTTONMODE.
NUMPTSALLOWEDBEFOREAPPROX	*<integer>*	When a frame's active zones contain this many points or less, the frame is not approximated, but always drawn in full. This applies to all frames when PLOTAPPROXIMATIONMODE is AUTOMATIC, and to the current frame only when PLOTAPPROXIMATIONMODE is NONCURRENTALWAYSAPPROX. This setting has no effect when PLOTAPPROXIMATIONMODE is set to ALLFRAMESALWAYSAPPROX.

Parameter Syntax		Notes
OKTOEXECUTESYSTEMCOMMAND	= *<boolean>*	Allow use of $!SYSTEM commands in macros. This is a security issue. If set to FALSE and the macro is run intermittantly you will be asked for permission to execute the $!SYSTEM command. If Tecplot is run in batch mode and this is FALSE an error will be generated and the macro will terminate.
OPENGLCONFIG		
{		
RUNDISPLAYLISTSAFTERBUILDING	= *<boolean>*	Tecplot defaults to building and running display lists simultaneously. Turn RunDisplayListsAfterBuilding on if you want to run the display lists after they are built. This may increase display list performance on some machines. The difference is often times neglegible.
ALLOWHWACCELERATION	= *<boolean>*	Windows only. This will disable hardware acceleration for Tecplot without having to change the Windows Display Properties. Setting ALLOWHWACCELERATION to NO may fix errors caused by hardware acceleration on buggy graphics card drivers.
SCREENRENDERING	= *<<renderconfig>>*	
IMAGERENDERING	= *<<renderconfig>>*	
MAXFILTERMAGNIFICATION	= *<integer>*	Sets the maximum magnification by non-texture resize filer before textures are used. This keeps Tecplot from creating textures which are too large. Default = 2.0. Setting this above three is not recomended, although setting below 1.0 will result in the use of a faster texture algorithm.
}		
PERCENTAGEOFPOINTSTOKEEP	= *<integer>*	Sets the percentage of points to keep in a frame when a frame is approximated. See the Tecplot User's Manual for a complete description.
PICKHANDLEWIDTH	*<op> <dexp>*	Value is in inches on the screen.

Parameter Syntax		Notes
PLOTAPPROXIMATIONMODE	= *<plotapproximationmode>*	Specifies the mode in which you want the plots to be approximated. See the Tecplot User’s Manual for a complete description of each mode.
PRINTDEBUG	= *<boolean>*	If TRUE, debugging information is sent to the standard output.
QUICKCOLORMODE	= *<quickcolormode>*	Choose objects for color changes made using the Quick Edit dialog.
ROTATION **{** **ROTATIONMODE** **CURRENTANGLE** **SMALLANGLE** **MEDIUMANGLE** **LARGEANGLE** **ROTATEDEGPERFRAMEUNIT** **SHOWGEOMS** **}**	 = *<rotationmode>* = *<op> <dexp>* = *<op> <dexp>* = *<op> <dexp>* = *<op> <dexp>* = *<integer>* = *<boolean>*	Settings for interactive rotations in 3-D.
ROTATEDEGPERFRAMEUNIT	= *<integer>*	
RULERPADDING	*<op> <dexp>*	Distance between workarea ruler and clipping edge for the paper and frames. Units are inches.
RULERTHICKNESS	*<op> <dexp>*	Value is in inches on the screen.
SCALE **{** **STEPSIZE** **SMALLSTEP** **MEDIUMSTEP** **LARGESTEP** **ZOOMSCALEPERFRAMEUNIT** **}**	 *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<op> <double>*	Settings for interactive scaling.
SCRBACKGROUNDCOLOR	= *<color>*	Set the workspace background color.
SECURESPOOLCOMMANDS	= *<boolean>*	Set to FALSE to allow $!SPOOLER commands outside the configuration file.
SHOWCONTINUOUSSTATUS	= *<boolean>*	
SHOWCOORDINATES	= *<boolean>*	
SHOWFRAMEBORDERSWHENOFF	= *<boolean>*	If TRUE, frame borders are drawn using a dashed line when they are turned off. This applies only to the screen and does not effect the hardcopy.

Parameter Syntax		Notes
SHOWSTATUSLINE	= *<boolean>*	
SHOWTEXTGEOMSINAPPROXVIEWS	= *<boolean>*	Set to TRUE if you want text and geometries to show up in frames using approximated plots
SHOWWAITDIALOGS	= *<boolean>*	If FALSE, all "Please Wait" and "Percent Done" dialogs will be disabled.
SOFTWARE3DRENDERING	= *<boolean>*	
TRACEREDRAWMODE	= *<traceredrawmode>*	
TRANSLATION **{** **STEPSIZE** **SMALLSTEP** **MEDIUMSTEP** **LARGESTEP** **}**	 *<op> <dexp>* *<op> <dexp>* *<op> <dexp>* *<op> <dexp>*	Settings for interactive translation.
UNIXHELPBROWSERCMD	= *<string>*	Sets the command used to launch a browser for add-ons that use HTML for their help file (UNIX only; Windows automatically connects to primary browser). For security reasons this command can only be used in the Tecplot configuration file.
USEAPPROXIMATEPLOTS	= *<boolean>*	Set to TRUE to use approximate plots. This will speed up any interactive rotations and translations, and many other actions as well.
USEDISPLAYLISTS	= *<boolean>*	
USEDOUBLEBUFFERING	= *<boolean>*	
USEDOUBLEFORDISPLAYLISTS	= *<boolean>*	
USEFASTAPPROXCONTINUOUSFLOOD	= *<boolean>*	
USEINITIALPLOTDIALOG	= *<boolean>*	Default is On.
USESTROKEFONTSFOR3DTEXT	= *<boolean>*	Use stroke fonts for data labels and ASCII scatter symbols in 3-D plots.
USESTROKEFONTSONSCREEN	= *<boolean>*	Set to TRUE to use Tecplot's internal stroke fonts, set to FALSE to use true type fonts. This option is only available under Windows.

Parameter Syntax		Notes
USETECPLOTPRINTDRIVERS	= *<boolean>*	This applies to Windows only. Set to TRUE to use Tecplot's printer drivers. Set to FALSE to use Windows printer drivers.
XORCOLOR	*<op> <integer>*	Color index to use for XORed lines. Set to 0 to make Tecplot calculate.
ZONEMAPNAMECOLUMNWIDTH	= *<double>*	Range is 10-1000. Sets the width of the Zone/Map Name column under Plot Attributes.

Example: This example does the following:

- Makes the frame borders show on the screen when they are turned off.
- Makes the middle mouse button be Redraw.
- Makes the right mouse button revert to Selector.
- Makes the default number of passes for smoothing 20.
- Turns off the status line.

```
$!INTERFACE
   SHOWFRAMEBORDERSWHENOFF = TRUE
   MOUSEACTIONS
   {
     MIDDLEBUTTON
     {
       BUTTONCLICK = REDRAW
     }
     RIGHTBUTTON
     {
       BUTTONCLICK = REVERTTOSELECT
     }
   }
   DATA
   {
    NUMSMOOTHPASSES = 20
   }
   SHOWSTATUSLINE = NO
```

$!INVERSEDISTINTERPOLATE

Syntax:

```
$!INVERSEDISTINTERPOLATE
   DESTINATIONZONE = <integer>
   [optional parameters]
```

Description: Interpolate selected variables from one or more zones onto a destination zone using the inverse distance method.

Required Parameter:

Parameters Syntax	Notes
`DESTINATIONZONE = `*<integer>*	Zone to interpolate to.

Optional Parameters:

Parameters Syntax	Default	Notes
`SOURCEZONES = `*<set>*	All zones except destination zone.	
`VARLIST = `*<set>*	All variables except spatial variables.	Choose the variables to interpolate. The spatial variables (X, Y and Z if 3-D) are not allowed.
`INVDISTEXPONENT = `*<dexp>*	`3.5`	
`INVDISTMINRADIUS = `*<dexp>*	`0.0`	
`INTERPPTSELECTION = `*<intrpptselection>*	`OCTANTNPOINTS`	
`INTERPNPOINTS = `*<integer>*	`8`	

Example: Interpolate variables 7-10 from zone 4 to zone 2:

```
$!INVERSEDISTINTERPOLATE
   SOURCEZONES     = [4]
   DESTINATIONZONE = 2
   VARLIST        = [7-10]
```

$!KRIG

Syntax:

```
$!KRIG
```

```
DESTINATIONZONE = <integer>
[optional parameters]
```

Description: Interpolate selected variables from a set of source zones to a destination zone using the kriging method.

Required Parameter:

Parameters Syntax	Notes
`DESTINATIONZONE =` *<integer>*	Zone to interpolate to.

Optional Parameters:

Parameters Syntax	Default	Notes
`SOURCEZONES =` *<set>*	All zones except the destination zone.	
`VARLIST =` *<set>*	All variables except spatial variables.	Choose the variables to interpolate. The spatial variables (X, Y and Z if 3-D) are not allowed.
`KRIGRANGE =` *<dexp>*	`0.3`	
`KRIGZEROVALUE =` *<dexp>*	`0.0`	
`KRIGDRIFT =` *<krigdrift>*	`LINEAR`	
`INTERPPTSELECTION =` *<interpptselection>*	`OCTANTNPOINTS`	
`INTERPNPOINTS` =*<integer>*	`8`	

Example: Krig from zones 3 and 4 onto zone 2. Only interpolate variable 7:

```
$!KRIG
   SOURCEZONES           = [3, 4]
   DESTINATIONZONE       = 2
   VARLIST               = [7]
```

$!LAUNCHDIALOG

Syntax: `$!LAUNCHDIALOG` *<dialogname>*
[no parameters]

Description: Launch a Tecplot interface dialog; *<dialogname>* can be one of ADVANCED3DCONTROL, AXISEDIT, COLORMAP, CONTOUR, CREATE1DLINE, CREATECIRCULARZONE, CREATERECTANGULARZONE, CREATEZONEFROMPOLYLINES, CREATEZONEFROMVALUES, CURVEINFO, DATAINFO, DATALABELS, DATASPREADSHEET, DELETEVARIABLES, DELETEZONES, DEPTHBLANKING, DUPLICATEZONE, EQUATION, EXPORT, EXTRACTCONTOURLINES, EXTRACTDISCRETEPOINTS, EXTRACTFEBOUNDARY, EXTRACTISOSURFACES, EXTRACTPOINTSFROMGEOMETRY, EXTRACTPOINTSFROMPOLYLINE, EXTRACTSLICEFROMPLANE, EXTRACTSLICES, EXTRACTSTREAMTRACES, EXTRACTSUBZONE, IJKBLANKING, IMPORT, INVERSEDISTANCEINTERPOLATION, ISOSURFACES, KRIGINGINTERPOLATION, LIGHTSOURCE, LINEARINTERPOLATION, LINEMAPLEGEND, LOADDATA, MACROPLAY, MACRORECORD, MACROVIEWER, MIRRORZONE, NEWLAYOUT, OPENLAYOUT, ORDERFRAMES, PAPERSETUP, POLARDRAWINGOPTIONS, PRINT, PROBEAT, PROBE, QUICKEDIT, QUICKMACROPANEL, RESET3DAXES, RGBCOLORLEGEND, RGBCOLORVARSANDRANGE, ROTATE2DDATA, RULERGRID, SAVEAS, SAVE, SCATTERLEGEND, SCATTERREFERENCESYMBOL, SCATTERSIZEANDFONT, SLICES, SMOOTH, SPATIALVARS, STREAMTRACES, STYLELINKING, THREEDAXISLIMITS, THREEDORIENTATIONAXIS, THREEDVIEWDETAILS, THREEDVIEWROTATE, TRANSFORMCOORDINATES, TRANSLATEMAGNIFY, TRIANGULATE, TWODDRAWORDER, VALUEBLANKING, VECTORARROWHEADS, VECTORLENGTH, VECTORREFERENCEVECTOR, VECTORVARS, WRITEDATA, ZONEMAPSTYLE. This command is mainly useful for the Tecplot demo.

Example: Launch Tecplot’s Macro Viewer dialog:

```
$!LAUNCHDIALOG MACROVIEWER
```

$!LIMITS

Syntax:

```
$!LIMITS
```

  [optional parameters]

Description: A SetValue command that sets some of the internal limits in Tecplot. See *Tecplot User’s Manual* for the default values for these limits. The **$!LIMITS** command can only be used in the Tecplot configuration file.

Optional Parameters:

Parameter Syntax		Notes
MAXPTSINALINE	*<op> <integer>*	Maximum number of points for geometry polylines.
MAXCHRSINTEXTLABELS	*<op> <integer>*	Maximum number of characters in text labels.
MAXNUMCONTOURLEVELS	*<op> <integer>*	Maximum number of contour levels.
MAXPREPLOTVARS	*<op> <integer>*	Maximum number of variables allowed in an ASCII data file loaded into Tecplot.

Parameter Syntax	Notes
MAXPREPLOTZONES *<op>* *<integer>*	Maximum number of zones allowed in an ASCII data file loaded into Tecplot.
MAXNUMPICKOBJECTS *<op>* *<integer>*	Maximum number of objects to pick.

Example: Increase the maximum number of contour levels allowed to 1,000:

```
$!LIMITS
   MAXNUMCONTOURLEVELS = 1000
```

$!LINEARINTERPOLATE

Syntax:

```
$!LINEARINTERPOLATE
   DESTINATIONZONE = <integer>
   [optional parameters]
```

Description: Interpolate selected variables from a set of source zones to a destination zone using linear interpolation. The source zones cannot be I-ordered. Values assigned to the destination zone are equivalent to the results of using the probe tool in Tecplot.

Required Parameter:

Parameters Syntax	Notes
DESTINATIONZONE = *<integer>*	Zone to interpolate to.

Optional Parameters:

Parameters Syntax	Default	Notes
SOURCEZONES = *<set>*	All zones except the destination zone.	
VARLIST = *<set>*	All variables except spatial variables.	Choose the variables to interpolate. The spatial variables (X, Y and Z if 3-D) are not allowed.

Example: Do linear interpolation from zones 2, 3 and 4 onto zone 7. Interpolate only variables 3-7:

```
$!LINEARINTERPOLATE
   SOURCEZONES     = [2-4]
   DESTINATIONZONE = 7
```

```
VARLIST          = [3-7]
```

$!LINEMAP

Syntax: `$!LINEMAP` *[<set>]*
[optional parameters]

Description: A SetValue command that assigns attributes for individual Line-mappings. The *<set>* parameter immediately following the **`$!LINEMAP`** command is optional. If *<set>* is omitted then the assignment is applied to all Line-mappings, otherwise the assignment is applied only to the Line-mappings specified in *<set>*.

Optional Parameters:

Parameter Syntax		Notes
NAME	= *<string>*	
ASSIGN		
{		
ZONE	= *<integer>*	
XAXISVAR	*<op> <integer>*	
YAXISVAR	*<op> <integer>*	
THETAAXISVAR	*<op> <integer>*	
RAXISVAR	*<op> <integer>*	
XAXIS	*<op> <integer>*	
YAXIS	*<op> <integer>*	
FUNCTIONDEPENDENCY	= *<functiondependency>*	
SHOWINLEGEND	= *[ALWAYS, NEVER,AUTO]*	
SORT	*<sortby>*	
SORTVAR	= *<integer>*	
}		
CURVES		
{		
CURVETYPE	= *<curvetype>*	
EXTENDEDNAME	= *<string>*	Only used by the Extended Curve-fit Add-on.
EXTENDEDSETTINGS	= *<string>*	Only used by the Extended Curve-fit Add-on.
USEWEIGHTVAR	= *<boolean>*	
NUMPTS	*<op> <integer>*	
POLYORDER	*<op> <integer>*	
WEIGHTVAR	= *<integer>*	
INDVARMIN	*<op> <dexp>*	
INDVARMAX	*<op> <dexp>*	
USEINDVARRANGE	= *<boolean>*	
CLAMPSPLINE	= *<boolean>*	
SPLINEDERIVATIVEATSTART	*<op> <dexp>*	
SPLINEDERIVATIVEATEND	*<op> <dexp>*	
}		

Parameter Syntax		Notes
SYMBOLS		
{		
SHOW	= <boolean>	
COLOR	= <color>	
FILLMODE	= <fillmode>	
FILLCOLOR	= <color>	
SIZE	<op> <dexp>	
LINETHICKNESS	<op> <dexp>	
SKIPPING	<op> <dexp>	
SKIPMODE	= <skipmode>	Skip can be by index or distance depending on SKIPMODE.
SYMBOLSHAPE	<<symbolshape>>	
}		
BARCHARTS		
{		
SHOW	= <boolean>	
COLOR	= <color>	
FILLMODE	= <fillmode>	
FILLCOLOR	= <color>	
SIZE	<op> <dexp>	
LINETHICKNESS	<op> <dexp>	
}		
LINES		
{		
SHOW	= <boolean>	
COLOR	= <color>	
LINEPATTERN	= <boolean>	
PATTERNLENGTH	= <color>	
LINETHICKNESS	<op> <dexp>	
}		
ERRORBARS		
{		
SHOW	= <boolean>	
VAR	= <integer>	
BARTYPE	= <errorbartype>	
COLOR	= <color>	
LINETHICKNESS	<op> <dexp>	
SKIPPING	<op> <dexp>	
SKIPMODE	= <skipmode>	Skip can be by index or distance depending on SKIPMODE.
SIZE	<op> <dexp>	
}		
INDICES		The indices parameter is used to restrict the range of data plotted (and which lines are plotted if the data is IJ- or IJK-ordered).
{		
IJKLINES	= <ijklines>	
IRANGE	<<indexrange>>	
JRANGE	<<indexrange>>	
KRANGE	<<indexrange>>	
}		
ASSIGN		
{		
SORT	<sortby>	
SORTVAR	= <integer>	
}		

Examples:

Example 1: Assign variable 1 to be on the X-axis and variable 4 to be on the Y-axis for Line-mapping number 7:

```
$!LINEMAP [7]
   ASSIGN
   {
    XAXISVAR = 1
    YAXISVAR = 4
   }
```

Example 2: Make Error Bars red for all Line-mappings:

```
$!LINEMAP
   ERRORBARS
   {
    COLOR = RED
   }
```

Example 3: Set Line-mappings 3-5 to draw a polynomial curve fit of order 5:

```
$!LINEMAP [3-5]
   CURVES
   {
    POLYORDER = 5
    CURVETYPE = CURVFIT
   }
   LINES
   {
    SHOW = YES
   }
```

$!LINEPLOTLAYERS

Syntax: `$!LINEPLOTLAYERS`
   [optional parameters]

Description: A SetValue command that turns on or off Line-plot layers.

Optional Parameters:

Parameter Syntax		Notes
SHOWLINES	= *<boolean>*	
SHOWSYMBOLS	= *<boolean>*	
SHOWBARCHARTS	= *<boolean>*	
SHOWERRORBARS	= *<boolean>*	Line-mapping must have an error bar variable assigned for this to have an effect.

Example: Turn on the symbols layer for Line-plots:

```
$!LINEPLOTLAYERS
   SHOWSYMBOLS = YES
```

$!LINKING

Syntax:

```
$!LINKING
```

[optional parameters]

Description: Link attributes in two or more frames so that changes to attributes of one frame effect all linked frames.

Optional Parameters:

Parameter Syntax	Notes

WITHINFRAME
{
LINKAXISSTYLE = *<boolean>*
LINKGRIDLINESTYLE = *<boolean>*
LINKLAYERLINECOLOR = *<boolean>*
LINKLAYERLINEPATTERN = *<boolean>*
}	

BETWEENFRAMES
{
LINKCONTOURLEVELS = *<boolean>*
LINKFRAMESIZEANDPOSITION = *<boolean>*
LINKXAXISRANGE = *<boolean>*
LINKYAXISRANGE = *<boolean>*
LINKPOLARVIEW = *<boolean>*
LINK3DVIEW = *<boolean>*
LINKGROUP = *<sminteger_t>*
LINKAXISPOSITION = *<boolean>*
LINKVALUEBLANKING = *<boolean>*
LINKSLICEPOSITIONS = *<boolean>*
LINKISOSURFACEVALUES = *<boolean>*
}	

Example: The following example will set the link attribute for all frames in the layout to LINK3DVIEW.

```
$!LOOP |NUMFRAMES|
$!LINKING BETWEENFRAME LINK3DVIEW = YES
$!FRAMECONTROL PUSHTOP
$!ENDLOOP
```

$!LOADADDON

Syntax:

```
$!LOADADDON <string>
INITFUNCTION = <string>
ADDONSTYLE = <addonstyle>
```

Description: Load an add-on into Tecplot. The <*string*> is the name of the add-on to load. See the *Tecplot User's Manual* for instructions on how to specify the add-on.

Optional Parameters:

Parameters Syntax	Default	Notes
INITFUNCTION = *<string>*	**InitTecAddOn**	Name of the function inside of the add-on that is used to initialize the add-on.
ADDONSTYLE= *<string>*	**V7Standard**	Style of the add-on to load. This can be either V7STANDARD or V7ACTIVEX.

Example: Load the Circle Stream add-on. It is a **V7STANDARD** add-on stored in a library named cstream.

```
$!LOADADDON "cstream"
```

$!LOADCOLORMAP

Syntax: **$!LOADCOLORMAP** <*string*>
[no parameters]

Description: Load a color map file. The <*string*> is the name of the file to load.

Example:

```
$!LOADCOLORMAP "mycolors.map"
```

$!LOOP...$!ENDLOOP

Syntax: **$!LOOP** <*integer*>
$!ENDLOOP

Description: Process macro commands in a loop. Within the loop you may access the current loop counter using the internal macro variable |**Loop**|. Loops may be nested up to 10 levels deep.

Example: Process macro commands 3 times over:

```
$!LOOP 3
   .
   .
   .
$!ENDLOOP
```

$!MACROFUNCTION...$!ENDMACROFUNCTION

Syntax:

```
$!MACROFUNCTION
    NAME = <string>
    [optional parameters]
    .
    .
    .
$!ENDMACROFUNCTION
```

Description: Define a macro function. All commands between a **$!MACROFUNCTION** and the **$!ENDMACROFUNCTION** are associated with the macro function **NAME**. These commands are not executed when they are defined but are executed when a **$!RUNMACROFUNCTION** command is processed. Parameters can be passed to a macro function. Use |***n***| to reference the *n*th parameter. (See **$!RUNMACROFUNCTION)**. To use the **KEYSTROKE** option, <Crtl>+M must be pressed initially.

Required Parameter:

Parameter Syntax	Notes
NAME = *<string>*	Name of the macro function.

Optional Parameter:

Parameter Syntax	Default	Notes
RETAIN = *<boolean>*	**FALSE**	Set this to **TRUE** if you want Tecplot to retain this macro function when the macro in which this macro function was defined terminates. If the macro function is retained then it can be called when another macro is loaded at a later time.
SHOWINMACROPANEL = *<boolean>*	**TRUE**	Used only for macro functions within the tecplot.mcr file. Set this to **FALSE** if you do not want Tecplot to include the macro function in Tecplot's Quick Macro Panel.
KEYSTROKE= *<char>*		Allows keyboard shortcuts

Example: Define a macro function that redraws the current frame *n* times when <Crtl>+M is hit and then the 'R' key is pressed, where *n* is passed to the macro function:

```
$!MACROFUNCTION
  NAME = "ABC"
  KEYSTROKE = "R"
$!LOOP |n|
$!REDRAW
```

```
$!ENDLOOP
$!ENDMACROFUNCTION
```

$!NEWLAYOUT

Syntax: `$!NEWLAYOUT`
[no parameters]

Description: Clear the current layout and start again. A blank default frame will be created for you.

Example: `$!NEWLAYOUT`

$!OPENLAYOUT

Syntax: `$!OPENLAYOUT` *<string>*
[optional parameters]

Description: Open and read in a new layout file. The *<string>* is the name of the file to open.

Optional Parameters:

Parameter Syntax	Default	Notes
`ALTDATALOADINSTRUCTIONS =` *<string>*	`Null`	Specify alternate data load instructions. Tecplot data files: This is a list of filenames to use as replacements for data files referenced in the layout file. Use `"` to enclose file names that contain spaces or the `+` symbol. By default, separate file names listed in the ALTDATALOADINSTRUCTIONS are assigned to successive data sets that are referenced within a layout file. If you have a data set that references multiple data files, use the plus symbol, +, to group file names. Non-Tecplot formats (including data being input via a data loader add-on): This is a list of instructions that are passed on to the loader.
`APPEND =` *<boolean>*	`FALSE`	Set to `FALSE` if you want Tecplot to delete the current layout prior to reading in the new one.

Examples:

Example 1: Open a new layout file called abc.lay and replace the data file referenced in the layout file with `t.plt:`

```
$!OPENLAYOUT "abc.lay"
```

```
   ALTDATALOADINSTRUCTIONS = "t.plt"
```

Example 2: Open a new layout file called multiframe.lay and replace the first data set with `t.plt` and the second data set with the two files, `a.plt` and `b.plt:`

```
$!OPENLAYOUT "multiframe.lay"
   ALTDATALOADINSTRUCTIONS = '"t.plt" "a.plt"+"b.plt"'
```

$!PAPER

Syntax: `$!PAPER`
  [optional parameters]

Description: A SetValue command that sets the paper characteristics.

Optional Parameters:

Parameter Syntax	Notes
BACKGROUNDCOLOR = *<color>*	
ISTRANSPARENT = *<boolean>*	
ORIENTPORTRAIT = *<boolean>*	
SHOWGRID = *<boolean>*	
SHOWPAPER = *<boolean>*	
SHOWRULER = *<boolean>*	
PAPERSIZE = *<papersize>*	
RULERSPACING = *<paperrulerspacing>*	
PAPERGRIDSPACING = *<papergridspacing>*	
PAPERSIZEINFO **{** **LETTER** *<<papersize>>* **DOUBLE** *<<papersize>>* **A3** *<<papersize>>* **A4** *<<papersize>>* **CUSTOM1** *<<papersize>>* **CUSTOM2** *<<papersize>>* **}**	
REGIONINWORKAREA *<<rect>>*	Specify rectangle that must fit within the workarea. Units are in inches (that is, in the paper coordinate system).

Example: This example does the following:

- Turns off the paper grid.
- Makes the paper size **CUSTOM1**.
- Makes the dimensions for **CUSTOM1** to be 4 by 5 inches.

```
$!PAPER
   SHOWGRID = NO
   PAPERSIZE = CUSTOM1
   PAPERSIZEINFO
   {
    CUSTOM1
    {
     WIDTH  = 4
     HEIGHT = 5
    }
   }
```

$!PAUSE

Syntax: **$!PAUSE** <*string*>
   [*no parameters****]***

Description: Stop execution of a macro and optionally display a dialog with a message. If <*string*> is set to **""** then no dialog is displayed and the user must click in the work area to continue.

Example: Pause and display the message **This is the first example plot**:

```
$!PAUSE "This is the first example plot."
```

$!PICK *[Required-Control Option]*

Description: The different commands in the **PICK** compound function family are described separately in the following sections.

The **PICK** compound functions are:

```
$!PICK ADD
   $!PICK ADDALL
   $!PICK ADDALLINRECT
   $!PICK CLEAR
   $!PICK COPY
   $!PICK CUT
```

```
$!PICK EDIT
$!PICK MAGNIFY
$!PICK PASTE
$!PICK POP
$!PICK PUSH
$!PICK SETMOUSEMODE
$!PICK SHIFT
```

$!PICK ADD

Syntax:

```
$!PICK ADD
   X = <dexp>
   Y = <dexp>
   [optional parameters]
```

Description: Attempt to pick an object at a specific location on the paper.

Required Parameters:

Parameters Syntax	Notes
X = *<dexp>*	X-location (in inches) relative to the left edge of the paper.
Y = *<dexp>*	Y-location (in inches) relative to the top edge of the paper.

Optional Parameters

Parameters Syntax	Default	Notes
COLLECTINGOBJECTS = *<boolean>*	**FALSE**	If **FALSE,** the list of picked objects is cleared before the attempt is made to add a new object.
DIGGINGFOROBJECTS = *<boolean>*	**FALSE**	If **TRUE**, attempt to pick objects below any currently picked objects at this location.
IGNOREZONEOBJECTS = *<boolean>*	**FALSE**	If **TRUE**, pick operations will ignore zones and pick objects such as slices, iso-surfaces and streamtraces.

Example: Attempt to add to the list of picked objects by picking at paper location (1.0, 7.0). Do not clear the list of picked objects before picking:

```
$!PICK ADD
   X  = 1.0
   Y  = 7.0
   COLLECTINGOBJECTS = TRUE
```

$!PICK ADDALL

Syntax:

```
$!PICK ADDALL
   [optional parameters]
```

Description: Add all objects of a certain type to the list of picked objects.

Optional Parameters

Parameters Syntax	Default	Notes
`SELECTTEXT = ` *<boolean>*	`FALSE`	Select all text objects in the current frame.
`SELECTGEOMS = ` *<boolean>*	`FALSE`	Select all geometry objects in the current frame.
`SELECTFRAMES = ` *<boolean>*	`FALSE`	Select all frames.
`SELECTSTREAMTRACES = ` *<boolean>*	`FALSE`	Select all streamtrace objects in the current frame.
`SELECTMAPS = ` *<boolean>*	`FALSE`	Select all line map objects in the current frame.
`SELECTZONES = ` *<boolean>*	`FALSE`	Select all zone objects in the current frame.

Example: Add all text and geometries in the current frame to the list of picked objects:

```
$!PICK ADDALL
   SELECTTEXT  = TRUE
   SELECTGEOMS = TRUE
```

$!PICK ADDALLINRECT

Syntax:

```
$!PICK ADDALLINRECT
   X1 = <dexp>
   Y1 = <dexp>
   X2 = <dexp>
   Y2 = <dexp>
   [optional parameters]
```

Description: Add objects defined within a specified region to the list of picked objects. The region is defined in terms of the paper coordinate system. Optional filters can be used to restrict the objects selected. The region is defined by the two corner points (X1, Y1) and (X2, Y2).

Required Parameters:

Parameters Syntax	Notes
X1 = *<dexp>*	X-location (in inches) relative to the left edge of the paper.
Y1 = *<dexp>*	Y-location (in inches) relative to the top edge of the paper.
X2 = *<dexp>*	X-location (in inches) relative to the left edge of the paper.
Y2 = *<dexp>*	Y-location (in inches) relative to the top edge of the paper.

Optional Parameters

Parameters Syntax	Default	Notes
SELECTTEXT = *<boolean>*	**FALSE**	Select all text objects in the specified region.
SELECTGEOMS = *<boolean>*	**FALSE**	Select all geometry objects in the specified region.
SELECTFRAMES = *<boolean>*	**FALSE**	Select all frame objects in the specified region.
SELECTSTREAMTRACES = *<boolean>*	**FALSE**	Select all streamtrace objects in the specified region.
SELECTMAPS = *<boolean>*	**FALSE**	Select all line map objects in the specified region.
SELECTZONES = *<boolean>*	**FALSE**	Select all zone objects in the specified region.
SELECTGRIDAREA = *<boolean>*	**FALSE**	Select the grid area in specified region
SELECTCONTOURLABELS = *<boolean>*	**FALSE**	Select all contour labels in specified region
COLORFILTER = *<color>*	Not used.[a]	Only objects of this color will be selected.
LINEPATTERNFILTER = *<linepattern>*	Not used.[a]	Only geometry objects with this line pattern will be selected.
FONTFILTER = *<font>*	Not used.[a]	Only text objects with this font will be selected.
GEOMFILTER = *<geomtype>*	Not used.[a]	Only geometry objects of this type will be selected.

[a] There is no default for this parameter. If this parameter is omitted then the corresponding filter is not used.

Example: Pick all circles using a dashed line pattern within the rectangle bounded by the points (0, 0) and (3, 5):

```
$!PICK ADDALLINRECT
   SELECTGEOMS              = TRUE
   LINEPATTERNFILTER        = DASHED
   GEOMFILTER               = CIRCLE
   X1                       = 0
   Y1                       = 0
```

```
X2                       = 3
Y2                       = 5
```

$!PICK CLEAR

Syntax: `$!PICK CLEAR`
[no parameters]

Description: Delete all objects that are currently picked. (These objects cannot be retrieved.)

Example: `$!PICK CLEAR`

$!PICK COPY

Syntax: `$!PICK COPY`
[no parameters]

Description: Copy all objects that are currently picked to the paste buffer.

Example: `$!PICK COPY`

$!PICK CUT

Syntax: `$!PICK CUT`
[no parameters]

Description: Copy all objects that are currently picked to the paste buffer and then delete them.

Example: `$!PICK CUT`

$!PICK EDIT

Syntax: `$!PICK EDIT`
[parameters]

Description: Perform a global edit operation on the currently picked objects. Only one edit operation is allowed per **`$!PICK EDIT`** command. Objects are edited only if the supplied parameter is relevant. Actions taken using the Quick Edit dialog in Tecplot generate these commands.

Parameters: Must select one from this table.

Parameters Syntax	Notes
ARROWHEADANGLE = *<dexp>*	Angle is in degrees.
ARROWHEADSIZE = *<dexp>*	Value is in Y-frame units (0-100).
LINETHICKNESS = *<dexp>*	Value is in Y-frame units (0-100).
PATTERNLENGTH = *<dexp>*	Value is in Y-frame units (0-100).
SIZE = *<dexp>*	Value is in Y-frame units. This applies to things like symbols.
TEXTHEIGHTBYPERCENT = *<dexp>*	Value is in Y-frame units (0-100).
TEXTHEIGHTBYPOINTS = *<dexp>*	Value is in points.
ARROWHEADATTACHMENT = *<arrowheadattachment>*	
ARROWHEADSTYLE = *<arrowheadstyle>*	
FONT = *<font>*	
GEOMSHAPE = *<geomshape>*	Applies only to scatter symbols or XY-plot symbols.
LINEPATTERN = *<linepattern>*	
OBJECTALIGN = *<objectalign>*	Only allowed if selected objects are all text and/or geometries.
TEXTCOLOR = *<color>*	
FILLCOLOR = *<color>*	
COLOR = *<color>*	
ASCIICHAR= *<symbolchar>*	
MESH {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.
MESH {MESHTYPE = *<meshplottype>***}**	Only operates on 2- or 3-D zone objects.
CONTOUR {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.
CONTOUR {CONTOURTYPE = *<contourplottype>}*	Only operates on 2- or 3-D zone objects.
VECTOR {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.
VECTOR {VECTORTYPE = *<vectorplottype>***}**	Only operates on 2- or 3-D zone objects.
SCATTER {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.
SCATTER {FILLMODE = *<fillmode>***}**	Only operates on 2- or 3-D zone objects.
SHADE {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.
SHADE {SHADETYPE = *<shadetype>***}**	Only operates on 2- or 3-D zone objects.
BOUNDARY {SHOW = *<boolean>***}**	Only operates on 2- or 3-D zone objects.

Parameters Syntax	Notes
`BOUNDARY {SUBBOUNDARY =` *<subboundary>*`}`	Only operates on 2- or 3-D zone objects.
`ERRORBARS {SHOW =` *<boolean>*`}`	Only operates on XY line mapping objects.
`ERRORBARS {BARTYPE =`*<errorbartype>*`}`	Only operates on XY line mapping objects.
`LINES {SHOW =` *<boolean>*`}`	Only operates on XY line mapping objects.
`BARCHARTS {SHOW =` *<boolean>*`}`	Only operates on XY line mapping objects.
`BARCHARTS {ISFILLED =` *<boolean>*`}`	Only operates on XY line mapping objects.
`SYMBOLS {SHOW =` *<boolean>*`}`	Only operates on line mapping objects.
`SYMBOLS {ISFILLED =` *<boolean>*`}`	Only operates on mapping objects.
`CURVES {CURVETYPE =` *<curvetype>*`}`	Only operates on XY line mapping objects.
`SHOWBORDER =` *<boolean>*	Only operates on frame objects.

Examples:

Example 1: Set all picked objects to use the color yellow:

```
$!PICK EDIT
   COLOR = YELLOW
```

Example 2: Set all picked objects to use the dashed line pattern:

```
$!PICK EDIT
   LINEPATTERN = DASHED
```

Example 3: Set all picked objects (which are zones) to use the contour plot type of flooding:

```
$!PICK EDIT
   CONTOUR {CONTOURTYPE = FLOOD}
```

$!PICK MAGNIFY

Syntax:

```
$!PICK MAGNIFY
   MAG = <dexp>
```

Description: Magnify all picked objects. The objects will also be translated proportional to the distance between their anchor position and the anchor position of the first object picked.

Example: Magnify all objects by 1.5:

```
$!PICK MAGNIFY
   MAG = 1.5
```

$!PICK PASTE

Syntax:

```
$!PICK PASTE
```

[no parameters]

Description: Paste the currently picked objects from the paste buffer to the work area.

Example:

```
$!PICK PASTE
```

$!PICK POP

Syntax:

```
$!PICK POP
```

[no parameters]

Description: Change the order in which objects are drawn by popping the currently picked objects to the front. Only frames, text, geometries, and the grid area for 2-D plots are allowed.

Example:

```
$!PICK POP
```

$!PICK PUSH

Syntax:

```
$!PICK PUSH
```

[no parameters]

Description: Change the order in which objects are drawn by pushing the currently picked objects back. Only frames, text, geometries, and the grid area for 2-D plots are allowed.

Example:

```
$!PICK PUSH
```

$!PICK SETMOUSEMODE

Syntax:

```
$!PICK SETMOUSEMODE
   MOUSEMODE = <mousemode>
```

Description: Prepare to pick objects by setting the mouse mode to **SELECT** or **ADJUST**. This command also clears the list of picked objects (that is, unpicks all picked objects).

Required Parameter:

Parameter Syntax	Notes
MOUSEMODE = *<mousemode>*	Set to **SELECT** or **ADJUST**.

Example: Set the mouse mode so picked objects are **adjusted**:

```
$!PICK SETMOUSEMODE
   MOUSEMODE = ADJUST
```

$!PICK SHIFT

Syntax:

```
$!PICK SHIFT
   X = <dexp>
   Y = <dexp>
   [optional parameters]
```

Description: Shift the currently picked objects. Objects are shifted relative to their starting position. X and Y shift amounts are in paper units (inches). If snapping is in effect then it is applied after shifting in X and Y. (See the SetValue commands **$!GLOBALFRAME SNAPTOGRID** and **$!GLOBALFRAME SNAPTOPAPER.**)

Required Parameters:

Parameters Syntax	Notes
X = *<dexp>*	Shift amount in the X-direction. Units are inches.
Y = *<dexp>*	Shift amount in the Y-direction. Units are inches.

Optional Parameter:

Parameters Syntax	Default	Notes
POINTERSTYLE = *<pointerstyle>*	**ALLDIRECTIONS**	Only frames and non-3-D grid area objects can use a pointer style that is not **ALLDIRECTIONS**.

Example: Shift the currently picked objects 1 inch to the right and 2 inches down:

```
$!PICK SHIFT
   X = 1
   Y = 2
```

$!PLOTTYPE

Syntax:

```
$!PLOTTYPE <plottype>
```

[no parameters]

Description: Changes plot types between valid Tecplot modes such as XYLine and Cartesian2D. Valid options shown below.

Required Parameters:

Parameter Syntax	Notes
PLOTTYPE *<plottype>*	

Example: **Change the plot style to show a polar plot**

```
$!PLOTTYPE POLARLINE
```

$!POLARAXIS

Syntax:

```
$!POLARAXIS
```

[optional parameters]

Description: A SetValue command that assigns attributes for axes in a polar frame.

Optional Parameters:

Parameter Syntax		Notes
THETAMODE	= *<thetamode>*	
THETAPERIOD	= *<double>*	
GRIDAREA	*<<areastyle>>*	
VIEWPORTPOSITION	*<<rect>>*	
VIEWPORTSTYLE	*<<areastyle>>*	
THETADETAIL	*<<axisdetail>>*	
RDETAIL	*<<axisdetail>>*	
PRECISEGRID	<<precisegrid>>	
PRESERVEAXISSCALE	<boolean>	

Example: Set the Theta range, in Radians, from Pi to -Pi.

```
$!POLARAXIS THETAMODE = RADIANS
$!POLARAXIS THETAPERIOD = 6.28318530718
$!POLARAXIS THETADETAIL{VALUEATORIGIN = 0}
$!POLARAXIS THETADETAIL{RANGEMIN = -3.14159265359}
```

$!POLARTORECTANGULAR

Syntax: **$!POLARTORECTANGULAR** *<set>*
[no parameters]

Description: Treat the variables currently assigned to X and Y as referring to R and θ and convert them to X and Y. In 3-D, X, Y and Z refer to R, θ, and ψ. Tecplot has addition capabilities for transforming coordinates, please see **$!TRANSFORMCOORDINATES.**

Example: Convert zones 1, 2 and 3 from polar to rectangular:

$!POLARTORECTANGULAR [1-3]

$!POLARVIEW

Syntax: `$!POLARVIEW`
[optional parameters]

Description: Sets the viewing style for polar plots in a layout.

Required Parameters:

Parameter Syntax		Notes
EXTENTS	= *<<rect>>*	View extents of transformed X & Y in polar plots. Numbers listed are in the form of grid units.

Example:

```
Set the view of the polar plot to view the full extents
   of the plot area.
$!POLARVIEW
 EXTENTS
 {
  X1=10
  Y1=10
  X2=90
  Y2=90
 }
```

$!PRINT

Syntax: `$!PRINT`
[no parameters]

Description: Print the current layout to a printer or send the print instructions to a file. Use the **`$!PRINTSETUP`** SetValue command to configure printing.

Example: `$!PRINT`

$!PRINTSETUP

Syntax: **$!PRINTSETUP**
[optional parameters]

Description: A SetValue command that sets the attributes for printing. Use **$!PRINT** to do the actual printing. See **$!EXPORTSETUP** and **$!EXPORT** if you intend to create image files destined for desktop publishing programs.

Optional Parameters:

Parameter Syntax	Notes
PRINTFNAME = *<string>*	Name of the file to write to if **SENDPRINTTOFILE** is **TRUE.**
PRECISION *<op> <integer>*	Applies only if **EXPORTFORMAT** is **HPGL2**, **PS**, **EPS**, or **RASTERMETAFILE**.
SENDPRINTTOFILE = *<boolean>*	If **TRUE** then **PRINTFNAME** is name of file to write to.
NUMHARDCOPYCOPIES *<op> <integer>*	Applies only when **DRIVER** = **PS.**
LARGEPAPEROK = *<boolean>*	Applies only when **DRIVER** = **HPGL**.
DRIVER = *<printerdriver>*	Only applies if using the Tecplot printer drivers. *See* **$!INTERFACE USETECPLOTPRINTDRIVERS**.
PALETTE = *<palette>*	Must choose options valid for current **DRIVER** setting.
PENSPEED *<op> <integer>*	
PLOTTERUNITSPERINCH *<op> <dexp>*	Applies only to **HPGL** and **HPGL2** output.
JOBCONTROL **{** **HPGLMOPUPSTR** = *<string>* **HPGL2MOPUPSTR** = *<string>* **POSTMOPUPSTR** = *<string>* **LGMOPUPSTR** = *<string>* **HPGLSETUPSTR** = *<string>* **HPGL2SETUPSTR** = *<string>* **POSTSETUPSTR** = *<string>* **LGSETUPSTR** = *<string>* **}**	These strings contain characters to be sent at the beginning and ending of a print file. These strings most often contain escape sequences used to switch modes on the printer. Non-printable characters can be inserted. Use **^***nnn* to insert a character with ordinal value *nnn*. Use \ to force the character after the \ to be inserted. Use **$B** for a Backspace, **$E** for Esc, **$C** for a carriage return, and **$X** for the Delete key.
SPOOLER **{** **HPGL2MONOSPOOLCMD** = *<string>* **HPGL2COLORSPOOLCMD** = *<string>* **HPGLSPOOLCMD** = *<string>* **PSMONOSPOOLCMD** = *<string>* **PSCOLORSPOOLCMD** = *<string>* **LGSPOOLCMD** = *<string>* **}**	These strings contain the system command needed to send a file to the print spooler on your computer. Use the **@** symbol as a place holder for where you normally insert the name of the file to be printed. For security reasons these commands can only be used in the Tecplot configuration file.

Parameter Syntax		Notes
PLOTTERPENMAP	= <<*plotterpenmap*>>	Assign plotter pens to objects or colors. See the *Tecplot User's Manual.*
USEISOLATIN1FONTS-INPS	= <*boolean*>	Use extended ISO-Latin1 fonts when generating PostScript output using Tecplot's internal PostScript driver.
FORCEEXTRA3DSORTING	= <*boolean*>	
NUMLIGHTSOURCESHADES	= <*integer*>	
IMAGERESOLUTION	= <*integer*>	
PRINTRENDERTYPE	= <*printrendertype*>	
RGBLEGENDOUTPUTRESOLUTION	= <*integer*>	Default=50. Determines the number of triangles which compose the bottom layer of the RGB Legend. This option is only availble through macro language (for example, the config file)

Example: This example does the following:

- Instruct Tecplot to send print output to the print spooler.
- Sets the spooler command for monochrome PostScript to be **lpr @**.
- Sets the print driver to be monochrome PostScript.

```
$!PRINTSETUP
   SENDPRINTTOFILE = FALSE
   DRIVER = PS
   PALETTE = MONOCHROME
   SPOOLER
   {
    PSMONOSPOOLCMD = "lpr @"
   }
```

$!PROMPTFORFILENAME

Syntax:

```
$!PROMPTFORFILENAME  <macrovar>
   DIALOGTITLE  = <string>
   DEFAULTFNAME = <string>
   FILEFILTER   = <string>
```

Description: Instruct Tecplot to launch a file selection dialog. The resulting file name will be placed in <*macrovar*>. If the user cancels out of the dialog then <*macrovar*> will be empty (see the example below).

Optional Parameter:

Parameter Syntax	Default	Notes
`DIALOGTITLE = `*<string>*	`Null`	Include a title at the top of the dialog.
`DEFAULTFNAME` = *<string>*	`Null`	Make the dialog come up with a default file name.
`FILEFILTER = `*<string>*	`Null`	Set the filter for the file selection dialog.
`FILEMUSTEXIST = `*<string>*	`TRUE`	

Example: Prompt the user for the name of a file to delete:

```
$!PROMPTFORFILENAME|filetodelete|
  DIALOGTITLE = "Delete File"
  FILEFILTER = "*.*"

$!IF "|filetodelete|" != ""
  $!IF |OPSys| = 1  # UNIX
    $!System "rm |filetodelete|"
  $!Endif
  $!IF |OPSys| = 2  # DOS
    $!System "del |filetodelete|"
  $!Endif
$!Endif
```

$!PROMPTFORTEXTSTRING

Syntax: `$!PROMPTFORTEXTSTRING` *<macrovar>*
`INSTRUCTIONS = `*<string>*

Description: Instruct Tecplot to launch a dialog containing a single line text field and optional instructions. The user enters text into the text field and the resulting string is assigned to *<macrovar>*.

Optional Parameter:

Parameter Syntax	Default	Notes
`INSTRUCTIONS = `*<string>*	`Null`	Include text at the top of the dialog to instruct the user regarding the value to enter. In Windows, this is limited to three lines of text.

Example:

```
$!PROMPTFORTEXTSTRING |timestring|
  INSTRUCTIONS = "Enter the time of the experiment"
```

$!PROMPTFORYESNO

Syntax:

```
$!PROMPTFORYESNO  <macrovar>
  INSTRUCTIONS = <string>
```

Description: Instruct Tecplot to launch a dialog containing two buttons, one labeled **Yes** and the other **No**. The *<macrovar>* is assigned the string **Yes** or **No** depending on the selection.

Optional Parameter:

Parameter Syntax	Default	Notes
INSTRUCTIONS = *<string>*	**Null**	Include text at the top of the dialog with instructions.

Example:

```
$!PROMPTFORYESNO |goforit|
  INSTRUCTIONS = "Do you want to go for it?"

$!IF "|goforit|" == "YES"
   ... code that goes for it....
$!ENDIF
```

$!PROPAGATELINKING

Syntax:

```
$!PROPAGATELINKING
    [optional parameters]
```

Description: Link multiple frames, ether within frame or between frames.

Optional Parameter:

Parameter Syntax		Notes

`LINKTYPE`	`= WITHINFRAME`
	or
	`BETWEENFRAMES`	

`FRAMECOLLECTION`	`= ALL` *or* `PICKED`	

Example:

```
$!PROPAGATELINKING
LINKTYPE = BETWEENFRAMES
   FRAMECOLLECTION = ALL
```

$!PUBLISH

Syntax: `$!PUBLISH` *<string>*

Description: Create an HTML file displaying one or more images. A linked layout with packaged data may be included. You must provide the file name.

Optional Parameter:

Parameter Syntax	Default	Notes
`INCLUDELAYOUTPACKAGE = ` *<boolean>*	`No`	Select YES to create a linked layout file.
`IMAGESELECTION = ` *<imagestyle>*	`ONEPERFRAME`	Selecting ONEPERFRAME will create one image per frame, selecting WORKSPACEONLY creates one image which includes all your frames.

Example:

```
$!PUBLISH "C:\TEC100\separate.html"
INCLUDELAYOUTPACKAGE = NO
   IMAGESELECTION = ONEPERFRAME
```

$!QUIT

Syntax: `$!QUIT`

Description: Terminate the execution of the Tecplot program.

Example: `$!QUIT`

$!RAWCOLORMAP

Syntax:

```
$!RAWCOLORMAP
   <colormaprawdata>
```

Description: Assign the RGB values that define the Raw user-defined color map. This does not set the color map to use the Raw user-defined color map. Use **`$!COLORMAP`** to set the current color map.

Required Parameter:

Parameter Syntax	Notes
<colormaprawdata>	This is a list of RGB values.

Example: Assign the Raw user-defined color map to a gray scale using 11 colors:

```
$!RAWCOLORMAP
RAWDATA
11
0             0        0
25            25       25
50            50       50
75            75       75
100           100      100
125           125      125
150           150      150
175           175      175
200           200      200
225           225      225
255           255      255
```

$!READDATASET

Syntax:

```
$!READDATASET <string>
   [optional parameters]
```

Description: Read one or more data files into Tecplot to form a new data set.

Optional Parameters:

Parameters Syntax	Default	Notes
IJKSKIP **{** **I =** *<integer>* **J =** *<integer>* **K =** *<integer>* **}**	 1 1 1	Use values greater than 1 to skip data points.
RESETSTYLE = *<boolean>*	**TRUE**	Set to **FALSE** if you want Tecplot to keep the current style. This only applies if **READDATA OPTION** is not **APPEND.**
INCLUDETEXT = *<boolean>*	**TRUE**	Set to **TRUE** to load in any text in the data files.
INCLUDEGEOM = *<boolean>*	**TRUE**	Set to **TRUE** to load in any geometries in the data files.
INCLUDECUSTOMLABELS = *<boolean>*	**TRUE**	Set to **TRUE** to load in any custom labels in the data files.
INCLUDEDATA = *<boolean>*	**TRUE**	Set to **TRUE** to load in any field data in the data files.
INITIALPLOTFIRSTZONEONLY = *<boolean>*		
INITIALPLOTTYPE = *<plottype>*		Allows faster performance for files with multiple zones.
DATASETREADER = *<string>*	**None.**	Used to specify an alternate data reader for Tecplot.
VARLOADMODE = *<varloadmode>*	**BYPOSITION**	Set to **BYPOSITION** to load variables based on their position in the file. Set to **BYNAME** to load variables based on their name. If set to **BYNAME**, then **VARNAMELIST** must be supplied as well.
VARNAMELIST = *<string>*	**None.**	Use this to list the names of the variables to load into Tecplot. Names separated by a ; or a + are joined together to form a set of aliases for a given variable.
VARPOSITIONLIST = *<set>*	All vars.	Use this to reduce the number of variables loaded.
ZONELIST = *<set>*	All zones.	Use this to reduce the number of zones loaded.
READDATAOPTION = *<readdataoption>*	**NEW**	Set to **APPEND** to append the new zones to the zones in the data set that existed prior to using this command. Set to **NEW** to remove the data set from the current frame prior to reading in the new data set. If other frames use the same data set they will continue to use the old one. Set to **REPLACE** to replace the data set attached to the current frame and to all other frames that use the same data set, with the new data set.
COLLAPSEZONESANDVARS = *<boolean>*	**FALSE**	Renumber zones and variables if zones or variables are disabled.

Examples:

Example 1: Read in the data files `t1.plt` and `t2.plt` to form a single data set in Tecplot:

```
$!READDATASET  "t1.plt t2.plt"
```

Example 2: Read in the datafile `t1.plt`. Only read in zones 1 and 4. Skip over every other I-index:

```
$!READDATASET  "t1.plt"
   ZONELIST = [1,4]
   IJKSKIP
   {
    I = 2
   }
```

Example 3: Read in the data files `t1.plt`, `t2.plt`, and `t3.plt`. Append the new data set to the current one:

```
$!READDATASET  "t1.plt t2.plt t3.plt"
   READDATAOPTION = APPEND
```

Example 4: Read in the data files `t1.plt` and `t2.plt` from directory `/users/john/testrun7/runb`:

```
$!VARSET |BASEDIR| = "/users/john/testrun7/runb"
   $!READDATASET "|basedir|/t1.plt |basedir|/t2.plt"
```

$!READSTYLESHEET

Syntax: `$!READSTYLESHEET` *<string>*
[optional parameters]

Description: Read in a stylesheet file. The *<string>* is the name of the file to read.

Optional Parameters:

Parameters Syntax	Default	Notes
`INCLUDETEXT =` *<boolean>*	`TRUE`	Set to `TRUE` to load in any text in the stylesheet file.
`INCLUDEGEOM =` *<boolean>*	`TRUE`	Set to `TRUE` to load in any geometries in the stylesheet file.
`INCLUDEPLOTSTYLE =` *<boolean>*	`TRUE`	Set to `TRUE` to process commands related to plot style (mesh color, vector type, and so on).
`INCLUDESTREAMPOSITIONS =` *<boolean>*	`TRUE`	Set to `TRUE` to read in streamtrace starting positions.

Parameters Syntax	Default	Notes
INCLUDEFRAMESIZEANDPOSITION = *<boolean>*	**FALSE**	Set to **TRUE** if you want the current frame to be sized and positioned exactly like the frame used to create the stylesheet.
MERGE = *<boolean>*	**FALSE**	Set to **FALSE** to reset all frame attributes back to their factory defaults prior to reading in the stylesheet.
INCLUDECONTOURLEVELS = *<boolean>*	**TRUE**	Set to **TRUE** to read in all contour levels.
INCLUDEAUXDATA= *<boolean>*	**TRUE**	Set to **TRUE** to read auxillary data.

Example: Read the stylesheet file `t.sty`. Do not read in any text or geometries:

```
$!READSTYLESHEET  "t.sty"
   INCLUDETEXT     = FALSE
   INCLUDEGEOM     = FALSE
```

$!REDRAW

Syntax: `$!REDRAW`
   [optional parameters]

Description: Redraw the current frame.

Optional Parameter:

Parameter Syntax	Default	Notes
DOFULLDRAWING = *<boolean>*	**TRUE**	Set to **FALSE** to draw only a "trace" of the data in the frame.

Example: `$!REDRAW`

$!REDRAWALL

Syntax: `$!REDRAWALL`
   [optional parameters]

Description: Redraw all frames.

Optional Parameter:

Parameter Syntax	Default	Notes
DOFULLDRAWING = *<boolean>*	**TRUE**	Set to **FALSE** to draw only a "trace" of the data in each frame.

Example: `$!REDRAWALL`

$!REMOVEVAR

Syntax: **$!REMOVEVAR** *<macrouserdefvar>*

Description: Remove a user-defined macro variable. This frees up space so another user-defined macro variable can be defined.

Example: Remove the macro variable |**ABC**|:

```
$!REMOVEVAR |ABC|
```

$!RENAMEDATASETVAR

Syntax:

```
$!RENAMEDATASETVAR
   VAR    = <integer>
   NAME   = <string>
   [no optional parameters]
```

Description: Rename a data set variable in Tecplot.

Required Parameters:

Parameter Syntax	Notes
VAR = *<integer>*	Specify the variable number.
NAME = *<string>*	Specify the new variable name.

Example: Rename variable 1 to be **Banana**:

```
$!RENAMEDATASETVAR
   VAR    = 1
   NAME = "Banana"
```

$!RENAMEDATASETZONE

Syntax:

```
$!RENAMEDATASETZONE
   ZONE  = <integer>
   NAME  = <string>
   [no optional parameters]
```

Description: Rename a data set zone in Tecplot.

Required Parameters:

Parameter Syntax	Notes
ZONE = *<integer>*	Specify the zone number.
NAME = *<string>*	Specify the new zone name.

Example: Rename zone 1 to be **Banana**:

```
$!RENAMEDATASETZONE
   ZONE = 1
   NAME = "Banana"
```

$!RESET3DAXES

Syntax:

```
$!RESET3DAXES
   [no parameters]
```

Description: Reset the ranges on the 3-D axes.

Example: `$!RESET3DAXES`

$!RESET3DORIGIN

Syntax:

```
$!RESET3DORIGIN
   [optional parameters]
```

Description: Reposition the rotation origin in 3-D to be at the specified location.

Optional Parameter:

Parameter Syntax	Notes
`ORIGINRESETLOCATION` = *<originresetlocation>*	

Example:

```
$!RESET3DORIGIN
  ORIGINRESETLOCATION = DATACENTER
```

$!RESET3DSCALEFACTORS

Syntax:

```
$!RESET3DSCALEFACTORS
```

[no parameters]

Description: Recalculate the scale factors for the 3-D axes. Aspect ratio limits are taken into account.

Example:

```
$!RESET3DSCALEFACTORS
```

$!RESETVECTORLENGTH

Syntax:

```
$!RESETVECTORLENGTH
```

[no parameters]

Description: Reset the length of the vectors. Tecplot will find the vector with the largest magnitude and set the scaling factor so it will appear on the screen using the length specified by **$!FRAMESETUP VECTDEFLEN**.

Example:

```
$!RESETVECTORLENGTH
```

$!ROTATE2DDATA

Syntax:

```
$!ROTATE2DDATA
   ANGLE = <dexp>
```

[optional parameters]

Description: Rotate field data in 2-D about any point.

Required Parameter:

Parameter Syntax	Notes
`ANGLE = ` *<dexp>*	Specify angle of rotation in degrees.

Optional Parameters:

Parameter Syntax	Default	Notes
`ZONELIST = ` *<set>*	All zones.	Zones to rotate.
`X = ` *<dexp>*	`0`	X-origin to rotate about.
`Y = ` *<dexp>*	`0`	Y-origin to rotate about.

Example: Rotate zone 3 30 degrees about the point (7, 2):

```
$!ROTATE2DDATA
   ANGLE    = 30
   ZONELIST = [3]
   X        = 7
   Y        = 2
```

$!ROTATE3DVIEW

Syntax:

```
$!ROTATE3DVIEW <rotateaxis>
   ANGLE = <dexp>
   [optional parameters]
```

Description: Do a 3-D rotation about a given axis. The *<rotateaxis>* must be supplied.

Required Parameter:

Parameter Syntax	Notes
`ANGLE = ` *<dexp>*	Angle to rotate (in degrees).

Optional Parameter:

Parameter Syntax	Notes
`ROTATEORIGINLOCATION = ` *<rotateoriginlocation>*	
`VECTORX = ` *<dexp>*	Required when rotate axis is **`ABOUTVECTOR.`**

Parameter Syntax		Notes
VECTORY	= *<dexp>*	Required when rotate axis is **ABOUTVECTOR.**
VECTORZ	= *<dexp>*	Required when rotate axis is **ABOUTVECTOR.**

Example:

```
$!ROTATE3DVIEW PSI
ANGLE = 10
```

$!RUNMACROFUNCTION

Syntax: **$!RUNMACROFUNCTION** *<string> [<macroparameterlist>]*

Description: Execute commands defined in a macro function. The **<***string***>** references the name of the macro function to run. If the macro requires parameters, then include them (within parentheses) after the macro name.

Example: Run macro function **XYZ** and pass the value 7 as the first parameter and the value 3.5 as the second parameter:

```
$!RUNMACROFUNCTION "XYZ" (7,3.5)
```

$!SAVELAYOUT

Syntax: **$!SAVELAYOUT** *<string>*
[optional parameters]

Description: Save the current layout to a file. You must supply the file name.

Optional Parameter:

Parameters Syntax	Default	Notes
USERELATIVEPATHS = *<boolean>*	**FALSE**	If **TRUE,** all files referenced in the layout file will use relative paths.
INCLUDEDATA = *<boolean>*	**FALSE**	If **TRUE,** a layout package file will be created. The extension .lpk is recommended.
INCLUDEPREVIEW = *<boolean>*	**TRUE**	Applies only if **INCLUDEDATA** is **TRUE**.

Example: Save the current layout to a file called `ex1.lay`:

```
$!SAVELAYOUT "ex1.lay"
```

$!SET3DEYEDISTANCE

Syntax:

```
$!SET3DEYEDISTANCE
EYEDISTANCE = <dexp>
```

Description: Sets the distance from the viewer to the plane of the current center of rotation.

Example:

```
$!SET3DEYEDISTANCE
EYEDISTANCE = 13.5
```

$!SETAUXDATA

Syntax:

```
$!SETAUXDATA
   AUXDATALOCATION  = [zone/var/dataset/frame/linemap]
   NAME = <string>
   VALUESTRING = <string>
   [optional parameters]
```

Description: Add Auxilary Data in the form of name/value pairs to zones, frames or datasets. The name must begin with an underscore or letter, and may be followed bu one or more underscore, period, letter, or digit characters.

Required Parameters:

Parameter Syntax		Notes
AUXDATALOCATION	= *zone/dataset/frame*	
NAME	= *<string>*	
VALUESTRING	= *<string>*	

Optional Parameters:

Parameter Syntax		Notes
`ZONE`	= *<integer>*	Only required if **`AUXDATALOCATION`** = **`zone`**
`VAR`	= *<integer>*	Only required if **`AUXDATALOCATION`** = **`var`**
`MAP`	= *<integer>*	Only required if **`AUXDATALOCATION`** = **`linemap`**

Example: Set the selected Auxilary Data to Zone 2.:

```
$!SETAUXDATA
   AUXDATALOCATION = zone
   ZONE = 2
   NAME = 'VARIABLE.DATA'
   VALUESTRING = 'WEST SECTOR'
```

$!SETDATASETTITLE

Syntax: **`$!SETDATASETTITLE`** *<string>*
[no optional parameters]

Description: **`Set the title for the current data set.`**

Example: **`$!SETDATASETTITLE "My data set"`**

$!SETFIELDVALUE

Syntax:

```
$!SETFIELDVALUE
   ZONE        = <integer>
   VAR         = <integer>
   INDEX       = <integer>
   FIELDVALUE  = <dexp>
   AUTOBRANCH  = <boolean>
   [no optional parameters]
```

Description: Specify a field value (data set value) at a specified point index. If the zone referenced is IJ- or IJK-ordered then the point index is calculated by treating the 2- or 3-D array as a 1-D array.

Required Parameters:

Parameters Syntax	**Notes**
ZONE = *<integer>*	
VAR = *<integer>*	
FIELDVALUE = *<dexp>*	
AUTOBRANCH = *<boolean>*	Affects shared variables only. If true, the specified zone will no longer share that variable with the other zones. If false, the variable will still be shared, and the change to the variable will be shown for all zones where it is shared.
INDEX = *<integer>*	

Example: A data set contains 2 zones and 3 variables. Zone 2 is dimensioned 5 by 3. Set the value for variable 3 at I-, J-location 2, 2 to be 37.5:

```
$!SETFIELDVALUE
   ZONE        = 2
   VAR         = 3
   INDEX       = 7
   FIELDVALUE  = 37.5
   AUTOBRANCH = TRUE
```

Note that the **INDEX** value was calculated using:

```
INDEX = I + (J-1)*|MAXI| + (K-1) * |MAXI| * |MAXJ|
       = 5*(2-1)+2
       = 7
```

$!SETSTYLEBASE

Syntax: **$!SETSTYLEBASE** *<stylebase>*

   [no parameters]

Description: Instruct Tecplot on how to initialize frame style values when a new frame is created. During normal operation, Tecplot bases the style of a new frame on the factory defaults plus any changes assigned in the Tecplot configuration file. Layout files and stylesheet files, however, rely on Tecplot basing new frames only on the factory defaults. This command is typically not used by the casual user.

Example: Set the style base for frames to use the factory defaults:

```
$!SETSTYLEBASE FACTORY
```

$!SHARECONNECTIVITY

Syntax:

```
$!SHARECONNECTIVITY
   SOURCEZONE    = <integer>
   DESTINATIONZONE = <integer>
   [no optional parameters]
```

Description: Share the nodemap between the source and destination zones, presuming that the zones are FE and have the same element type and number of nodes.

Required Parameters:

Parameter Syntax		Notes
SOURCEZONE	= *<integer>*	
DESTINATIONZONE	= *<integer>*	

Example: Shares the conectivity of the second zone with the sixth zone.:

```
$!SHARECONNECTIVITY
   SOURCEZONE   = 2
   DESTINATIONZONE = 6
```

$!SHAREFIELDDATAVAR

Syntax:

```
$!SHAREFIELDDATAVAR
   SOURCEZONE   = <integer>
   VAR = <integer>
   DESTINATIONZONE = <integer>
   [no optional parameters]
```

Description: Allows sharing of the specified variable from the soure zone to the destination zone. Zone must be of the same type (ordered or FE) and dimensions. Cell centered variables in FE must have the same number of cells. Sharing is not allowed if either zone has global face neighbors.

Required Parameters:

Parameter Syntax		Notes
`SOURCEZONE`	= *<integer>*	
`VAR`	= *<integer>*	
`DESTINATIONZONE`	= *<integer>*	

Example: Shares the third variable from the second zone, with the fifth zone:

```
$!SHAREFIELDDATAVAR
   SOURCEZONE  = 2
   VAR   = 3
   DESTINATIONZONE = 5
```

$!SHIFTLINEMAPSTOBOTTOM

Syntax: **`$!SHIFTLINEMAPSTOBOTTOM`** *<set>*
   [**no parameters**]

Description: Shift a list of Line-mappings to the bottom of the Line-mapping list. This in effect causes the selected Line-mappings to be drawn last.

Example: Shift Line-mappings 2 and 4 to the bottom:

```
$!SHIFTLINEMAPSTOBOTTOM [2,4]
```

$!SHIFTLINEMAPSTOTOP

Syntax: **`$!SHIFTLINEMAPSTOTOP`** *<set>*
   [**no parameters**]

Description: Shift a list of Line-maps to the top of the Line-map list. This in effect causes the selected Line-maps to be drawn first.

Example: Shift Line-maps 2 and 4 to the top:

```
$!SHIFTLINEMAPSTOTOP [2,4]
```

$!SHOWMOUSEPOINTER

Syntax: **$!SHOWMOUSEPOINTER** *<boolean>*
[optional parameters]

Description: The mouse icon may be deactived within a macro to enhance the on-screen animation. It must be reactivated before exiting the macro.

Example:

```
$!SHOWMOUSEPOINTER NO
$!LOOP 36
  $!ROTATE3DVIEW X
    ANGLE = 5
  $!REDRAW
$!ENDLOOP
$!SHOWMOUSEPOINTER YES
```

$!SKETCHAXIS

Syntax: **$!SKETCHAXIS**
[optional parameters]

Description: A SetValue command that assigns attributes for axes in a sketch mode frame. Axes are rarely used in sketch frames.

Optional Parameters:

Parameter Syntax		Notes
DEPXTOYRATIO	*<op> <dexp>*	**AXISMODE** must be **XYDEPENDENT** to use this.
AXISMODE	**=** *<axismode>*	Set to **INDEPENDENT** or **XYDEPENDENT.**
GRIDAREASTYLE	*<<gridarea>>*	
XDETAIL	*<<axisdetail>>*	
YDETAIL	*<<axisdetail>>*	
PRECISEGRID	<<precisegrid>>	
VIEWPORTTOPSNAPTARGET	**=** *<integer>*	Default = 100
VIEWPORTTOPSNAPTOLERANCE	**=** *<integer>*	Default = 10

Parameter Syntax	Notes
PRESERVEAXISSCALEWHENRANGEISCHANGED = *<boolean>*	
AUTOADJUSTRANGESTONICEVALEUS = *<boolean>*	
VIEWPORTPOSITION = *<<rect>>*	
VIEWPORTNICEFITBUFFER = *<double>*	

Example: Change the axis mode to be **INDEPENDENT** for sketch mode in the current frame:

```
$!SKETCHAXIS
   AXISMODE = INDEPENDENT
```

$!SMOOTH

Syntax:

```
$!SMOOTH
   ZONE = <set>
   VAR  = <set>
   [optional parameters]
```

Description: Smooth data (reduce the spikes) for selected variables in selected zones.

Required Parameters:

Parameter Syntax	Notes
ZONE = *<set>*	Zones to smooth.
VAR = *<set>*	Variables to smooth. These cannot be X or Y if in 2-D or Z if in 3-D and they must be a dependent variable in XY-plots.

Optional Parameters:

Parameter Syntax	Default	Notes
NUMSMOOTHPASSES = *<integer>*	1	
SMOOTHWEIGHT = *<dexp>*	0.8	
SMOOTHBNDRYCOND = *<boundarycondition>*	FIXED	

Example: Smooth variables 3 and 4 in zone 2:

```
$!SMOOTH
   ZONE = [2]
   VAR  = [3,4]
```

$!STREAMTRACE *[Required-Control Option]*

Description: The different commands in the **STREAMTRACE** compound function family are described separately in the following sections.

The **STREAMTRACE** compound function family is:

```
$!STREAMTRACE ADD
   $!STREAMTRACE DELETALL
   $!STREAMTRACE DELETERANGE
   $!STREAMTRACE RESETDELTATIME
   $!STREAMTRACE SETTERMINATIONLINE
```

$!STREAMTRACE ADD

Syntax:

```
$!STREAMTRACE ADD
   [optional parameters]
```

Description: Add a single streamtrace or a rake of streamtraces to the current frame. The frame must be a 2-D or 3-D field plot.

Optional Parameters:

Parameters Syntax	Default	Notes
NUMPTS = *<integer>*	**1**	Use 1 to add a single streamtrace. Use *n*, *n*>1 for a rake of streamtraces.
STREAMTYPE = *<streamtype>*	a	
DIRECTION = *<streamdirection>*	**FORWARD**	

Parameters Syntax	Default	Notes

STARTPOS
{
 X = *<dexp>*
 Y = *<dexp>*
 Z = *<dexp>*
}
	
	0.0
	0.0
	0.0
		Z is necessary only if dealing with a 3-D streamtrace.

ALTSTARTPOS
{
 X = *<dexp>*
 Y = *<dexp>*
 Z = *<dexp>*
}
		This is required if **NUMPTS** is greater than 1 or if the streamtype is a volume rod or volume ribbon.

a. Tecplot determines the default streamtype based on a number of factors. It is best to always supply this parameter.

Example 1: Add a rake of 5 streamtraces in a 2-D field plot:

```
$!STREAMTRACE ADD
   NUMPTS     = 5
   STREAMTYPE = TWODLINE
   STARTPOS
   {
    X = 0.5
    Y = 0.5
   }
   ALTSTARTPOS
   {
    X = 0.5
    Y = 1.5
   }
```

Example 2: Add a single volume ribbon. Start the ribbon oriented parallel to the Z-axis:

```
$!STREAMTRACE ADD
   STREAMTYPE = VOLUMERIBBON
   STARTPOS
   {
    X = 3.0
    Y = 4.0
    Z = 1.0
   }
   ALTSTARTPOS
   {
    X = 3.0
    Y = 4.0
```

```
 Z = 8.0
}
```

$!STREAMTRACE DELETEALL

Syntax: `$!STREAMTRACE DELETEALL`

[no parameters]

Description: Deletes all streamtraces in the current frame. If the frame mode is 2-D, all 2-D streamtraces are deleted. If the frame mode is 3-D, all 3-D streamtraces are deleted.

Example:

```
$!STREAMTRACE DELETEALL
```

$!STREAMTRACE DELETERANGE

Syntax: `$!STREAMTRACE DELETERANGE`

[optional parameters]

Description: Delete a range of streamtraces. Streamtraces are numbered sequentially in the order they were created.

Optional Parameters:

Parameters Syntax	Default	Notes
`RANGESTART` = *<integer>*	1	
`RANGEEND` = *<integer>*	1	

Example: Delete streamtraces 3-5:

```
$!STREAMTRACE DELETERANGE
   RANGESTART   = 3
   RANGEEND     = 5
```

$!STREAMTRACE RESETDELTATIME

Syntax: `$!STREAMTRACE RESETDELTATIME`

[no parameters]

Description: Reset the time delta for dashed streamtraces. The delta time is reset such that a stream dash in the vicinity of the maximum vector magnitude will have a length approximately equal to 10 percent of the frame width.

Example: $!STREAMTRACE RESETDELTATIME

$!STREAMTRACE SETTERMINATIONLINE

Syntax:

```
$!STREAMTRACE SETTERMINATIONLINE
   <xyrawdata>
```

Description: Set the position of the termination line for streamtraces.

Required Parameter:

Parameters Syntax	Notes
<xyrawdata>	In 3-D, the termination line is defined in the eye coordinate system.

Example: Set the termination line using 3 points:

```
$!STREAMTRACE SETTERMINATIONLINE
   RAWDATA
   3
   4.0          7.0
   5.0          9.0
   5.0          3.0
```

$!SYSTEM

Syntax: `$!SYSTEM` *<string>*
[optional parameters]

Description: Instruct Tecplot to submit a command to the operating system. For security reasons, execution of the `$!SYSTEM` command can be disabled to prevent unauthorized execution of system commands via macros. Use the `OKTOEXECUTESYSTEMCOMMAND` option to the `$!INTERFACE` macro command.

Example: Submit the system command to copy the file `t7.plt` to `xxx.plt` (UNIX):

```
$!SYSTEM "cp t7.plt xxx.plt"
```

Optional Parameters:

Parameter Syntax	Default	Notes
WAIT = *<boolean>*	**TRUE**	*If* **TRUE***, Tecplot will wait until the execution of the system command has completed before continuing.*

$!THREEDAXIS

Syntax: **$!THREEDAXIS**
[optional parameters]

Description: A SetValue command that assigns attributes for axes in a 3-D frame.

Optional Parameters:

Parameter Syntax		Notes
XYDEPXTOYRATIO	*<op> <dexp>*	**AXISMODE** must be **XYDEPENDENT** to use this.
DEPXTOYRATIO	*<op> <dexp>*	**AXISMODE** must be **DEPENDENT** to use this.
DEPXTOZRATIO	*<op> <dexp>*	**AXISMODE** must be **DEPENDENT** to use this.
AXISMODE	= *<axismode>*	Set to **INDEPENDENT**, **XYDEPENDENT**, or **XYZDEPENDENT**.
ASPECTRATIOLIMIT	*<op> <dexp>*	Restrict the aspect ratio of the data.
ASPECTRATIORESET	*<op> <dexp>*	Set aspect ratio for the data to this value when **ASPECTRATIOLIMIT** is exceeded.
BOXASPECTRATIOLIMIT	*<op> <dexp>*	Restrict the aspect ratio of the axis box.
BOXASPECTRATIORESET	*<op> <dexp>*	Set aspect ratio for the axis box to this value when **ASPECTRATIOLIMIT** is exceeded.
EDGEAUTORESET	= *<boolean>*	Make Tecplot automatically choose edges to label.
FRAMEAXIS **{** **SHOW** **SIZE** **LINETHICKNESS** **COLOR** **XYPOS** **}**	 = *<boolean>* *<op> <dexp>* *<op> <dexp>* = *<color>* *<<xy>>*	
GRIDAREA	*<<gridarea>>*	
XDETAIL	*<<axisdetail>>*	
YDETAIL	*<<axisdetail>>*	

Parameter Syntax		Notes
ZDETAIL	*<<axisdetail>>*	
PRESERVEAXISSCALEWHENRANGEISCHANGED	= *<boolean>*	

Example: This example does the following:

- Changes the variable assigned to the Z-axis to be variable number 2.
- Turns off auto edge assignment and make axis labeling for the Y-axis occur on edge 2.

```
$!THREEDAXIS
   ZVAR = 2
   EDGEAUTORESET = FALSE
   YEDGE = 2
```

$!THREEDVIEW

Syntax:

```
$!THREEDVIEW
```

[optional parameters]

Description: A SetValue command that changes global attributes associated with the 3-D view.

Optional Parameters:

Parameter Syntax		Notes
DRAWINPERSPECTIVE	= *<boolean>*	
PSIANGLE	*<op> <dexp>*	Angle is in degrees.
THETAANGLE	*<op> <dexp>*	Angle is in degrees.
ALPHAANGLE	*<op> <dexp>*	Angle is in degrees.
FIELDOFVIEW	*<op> <dexp>*	
VIEWWIDTH	*<op> <dexp>*	
VIEWERPOSITION	= *<<xyz>>*	

Example: This example does the following:

- Switches to perspective.
- Changes the field of view.
- Rotates around psi by 20 degrees..

- Changes the viewer position.

```
$!THREEDVIEW
   DRAWNINPERSPECTIVE = YES
   FIELDOFVIEW = 100
   PSIANGLE += 20
   VIEWERPOSITION
   {
     X = 1.26
     Y = 1.25
     Z = 0.74
    }
```

$!TRANSFORMCOORDINATES

Syntax:

```
$!TRANSFORMCOORDINATES
   TRANSFORMATION=<transformation>
```

[optional parameters]

Description: Transforms all points in one or more zones from one coordinate system to another.

Required Parameter

Parameters Syntax	Notes
TRANSFORMATION = *<transformation>*	Transformation.

Optional Parameters:

Parameter Syntax	Default	Notes
CREATENEWVARIABLES = *<boolean>*	**FALSE**	If **TRUE**, then new variables X,Y,Z will be created if converting to rectangular coordinates, or R,THETA,PHI if converting to spherical. If **FALSE**, then you must specify the output variables.
THETAVAR = *<integer>*	**NONE**	Theta variable number. REQUIRED if the transformation is polar to rectangular or spherical to rectangular or if **CREATENEWVARIABLES** is **FALSE**.

Parameter Syntax	Default	Notes
RVAR = *<integer>*		R variable number. REQUIRED if the transformation is polar to rectangular or spherical to rectangular or if **CREATENEWVARIABLES** is **FALSE**.
PSIVAR = *<integer>*		PSI variable number. REQUIRED if the transformation is spherical to rectangular or if **CREATENEWVARIABLES** is **FALSE**.
XVAR = *<integer>*		X variable number. REQUIRED if the transformation is rectangular to polar or rectangular to spherical or **CREATENEWVARIABLES** is **FALSE**.
YVAR = *<integer>*		Y variable number. REQUIRED if the transformation is rectangular to polar or rectangular to spherical or **CREATENEWVARIABLES** is **FALSE**.
ZVAR = *<integer>*		Z variable number. REQUIRED if the transformation or rectangular to spherical or **CREATENEWVARIABLES** is **FALSE**.
ANGLESPEC = *<anglespec>*	**RADIANS**	Specifies whether data is in degrees or radians
ZONESET = *<set>*	*all zones*	Set if zones to operate on.

Example: Transform data from rectangular coordinates to polar coordinates specifying angles in degrees and creating new variables.

```
$!TRANSFORMCOORDINATES
  TRANSFORMATION = RECTTOPOLAR
  ANGLESPEC = DEGREES
  CREATENEWVARIABLES = YES
  XVAR = 2
  YVAR = 3
```

$!TRIANGULATE

Syntax:

```
$!TRIANGULATE
```

[optional parameters]

Description: Create a new zone by forming triangles from data points in existing zones.

Optional Parameters:

Parameters Syntax	Default	Notes
SOURCEZONES = *<set>*	All zones.	
USEBOUNDARY = *<boolean>*	**FALSE**	Specify one or more I-ordered zones that define boundaries across which no triangles can be created.
BOUNDARYZONES = *<set>*		Required if **USEBOUNDARY** is **TRUE.**
INCLUDEBOUNDARYPTS = *<boolean>*	**FALSE**	Set to **TRUE** if you also want the boundary points to be used to create triangles.
TRIANGLEKEEPFACTOR = *<dexp>*	**0.25**	

Example: Create a zone by triangulating data points from zones 1 and 2:

```
$!TRIANGULATE
   SOURCEZONES    = [1,2]
```

$!TWODAXIS

Syntax:

```
$!TWODAXIS
   [optional parameters]
```

Description: A SetValue command that assigns attributes for axes in a 2-D frame.

Optional Parameters:

Parameter Syntax		Notes
DEPXTOYRATIO	*<op> <dexp>*	**AXISMODE** must be **XYDEPENDENT** to use this.
AXISMODE	= *<axismode>*	Set to **INDEPENDENT** or **XYDEPENDENT**.
GRIDAREA	*<<gridarea>>*	
XDETAIL	*<<axisdetail>>*	
YDETAIL	*<<axisdetail>>*	
PRECISEGRID	<<precisegrid>>	
VIEWPORTTOPSNAPTARGET	= *<integer>*	Default = 100
VIEWPORTTOPSNAPTOLERANCE	= *<integer>*	Default = 10
VIEWPORTPOSTITION	<<rect>>	
VIEWPORTNICEFITBUFFER	= *<double>*	

Parameter Syntax	Notes
AUTOADJUSTRANGESTONICEVALUES = *<boolean>*	
PRESERVEAXISSCALEWHENRANGEISCHANGED = *<boolean>*	

Example: Set the X-axis to use variable 3 for a 2-D plot:

```
$!TWODAXIS
   XDETAIL {VARNUM = 3}
```

$!VARSET

Syntax: **$!VARSET** *<macrovar> < op> <dexp>*

[no parameters]

or

$!VARSET *<macrovar>* = *<string>*

[no parameters]

Description: Assign a value to a macro variable. If the macro variable did not exist prior to this command, then it is defined here. A macro variable can be assigned a value or a string.

Examples:

Example 1: Set the macro variable |**myvar**| to 3:

```
$!VARSET |myvar| = 3
```

Example 2: Add 2 to the macro variable |**myvar**|**:**

```
$!VARSET |myvar| += 2
```

Example 3: Set the macro variable |**File1**| to be **myfile.plt:**

```
$!VARSET |File1| = "myfile.plt"
```

Example 4: Set the macro variable |**F1**| to equal |**V2**| **+** |**V3**|, where |**V2**| and |**V3**| are predefined variables:

```
$!VARSET|V2| = 4
$!VARSET|V3| = 5
$!VARSET|F1| = (|V2| + |V3|)
```

$!VIEW *[Required-Control Option]*

Description: The different commands in the **VIEW** compound function family are described separately in the following sections.

The **VIEW** compound function family is:

```
$!VIEW AXISFIT
$!VIEW AXISMAKECURRENTVALUESNICE
$!VIEW AXISNICEFIT
   $!VIEW CENTER
   $!VIEW COPY
   $!VIEW DATAFIT
   $!VIEW FIT
   $!VIEW LAST
$!VIEW MAKECURRENTVIEWNICE
$!VIEW NICEFIT
   $!VIEW PASTE
   $!VIEW PUSH
$!VIEW RESETTOENTIRECIRCLE
   $!VIEW SETMAGNIFICATION
   $!VIEW TRANSLATE
   $!VIEW ZOOM
```

$!VIEW AXISFIT

Syntax: **$!VIEW AXISFIT**
[optional parameters]

Description: Reset the range on a specific axis so that it equals the minimum and maximum of the data being plotted. If the axis dependency is not independent then this action may also affect the range on another axis.

Optional Parameters:

Parameters Syntax	Default	Notes
AXIS = <*xyaxis*>	**'X'**	Default is **'T'** for polar plot type.
AXISNUM = <*integer*>	**1**	Only XY frame mode allows for this to be a number greater than 1.

Example: Reset the range on the Y-axis to fit the data being plotted:

```
$!VIEW AXISFIT
   AXIS = 'Y'
```

$!VIEW AXISMAKECURRENTAXISVALUESNICE

Syntax:

```
$!VIEW AXISMAKECURRENTAXISVALUESNICE
```

   [optional parameters]

Description: Reset the axis-line label values such that all currently displayed values are set to have the smallest number of significant digits possible.

Optional Parameters:

Parameters Syntax	Default	Notes
`AXIS = ` *<xyaxis>*	`'X'`	Default is `'T'` for polar plot type.
`AXISNUM = ` *<integer>*	`1`	Only XY line plots allow for this to be a number greater than 1.

Example: Set the range on the Z-axis to have nice values for the axis labels :

```
$!VIEW AXISMAKECURRENTAXISVALUESNICE
   AXIS = 'Z'
```

$!VIEW AXISNICEFIT

Syntax:

```
$!VIEW AXISNICEFIT
```

   [optional parameters]

Description: Reset the range on a specific axis so that it equals the minimum and maximum of the data being plotted, but makes the axis values "nice" by setting labels to have the smallest number of significant digits possible. If the axis dependency is not independent then this action may also affect the range on another axis.

Optional Parameters:

Parameters Syntax	Default	Notes
`AXIS = `*<xyaxis>*	`'X'`	*Default is* `'T'` *for polar plot type.*
`AXISNUM = `*<integer>*	`1`	Only XY frame mode allows for this to be a number greater than 1.

Example: Reset the range on the Y-axis to fit the data being plotted, with nice values on the axis-line:

```
$!VIEW AXISNICEFIT
   AXIS = 'Y'
```

$!VIEW CENTER

Syntax:

```
$!VIEW CENTER
```

[no parameters]

Description: Center the data within the axis grid area.

Example:

```
$!VIEW CENTER
```

$!VIEW COPY

Syntax:

```
$!VIEW COPY
```

[no parameters]

Description: Copy the current view to the view paste buffer. See also `$!VIEW PASTE.`

Example:

```
$!VIEW COPY
```

$!VIEW DATAFIT

Syntax:

```
$!VIEW DATAFIT
```

[no parameters]

Description: Fit the current set of data zones or line mappings being plotted within the grid area. This does not take into consideration text or geometries.

Example: `$!VIEW DATAFIT`

$!VIEW FIT

Syntax:

```
$!VIEW FIT
```

[no parameters]

Description: Fit the entire plot to the grid area. This also takes into consideration text and geometries that are plotted using the grid coordinate system. In 3-D, this also includes the axes.

Example: `$!VIEW FIT`

$!VIEW LAST

Syntax:

```
$!VIEW LAST
```

[no parameters]

Description: Retrieve the previous view from the view stack. Each frame mode within each frame maintains its own view stack. `$!VIEW LAST` will not reverse alterations to data.

Example: `$!VIEW LAST`

$!VIEW MAKECURRENTVIEWNICE

Syntax:

```
$!VIEW MAKECURRENTVIEWNICE
```

[no parameters]

Description: Shifts axis to make axis-line values nice without changing the extents of the window. Only works in Sketch/XY/2D.

Example: `$!VIEW MAKECURRENTVIEWNICE`

$!VIEW NICEFIT

Syntax: `$!VIEW NICEFIT`

[no parameters]

Description: Change view to make the extents of the frame neatly hold the plot with integer values for axis labels.. Only works in Sketch/XY/2D.

Example:

```
$!VIEW NICEFIT
```

$!VIEW PASTE

Syntax: `$!VIEW PASTE`

[no parameters]

Description: Retrieve the view from the view paste buffer and assign it to the current frame.

Example:

```
$!VIEW PASTE
```

$!VIEW PUSH

Syntax: `$!VIEW PUSH`

[no parameters]

Description: Instruct Tecplot to push the current view onto the view stack. A view will not be pushed if the current view is the same as the top view on the stack. Note that commands **`VIEW AXISFIT, VIEW CENTER, VIEW DATAFIT, VIEW FIT`**, and **`VIEW ZOOM`** automatically push a view onto the stack. Tecplot automatically pushes the current view onto the stack when a **`$!REDRAW`** command is issued and the current view is different from the top view on the view stack.

Example:

```
$!VIEW PUSH
```

$!VIEW RESETTOENTIRECIRCLE

Syntax:

```
$!VIEW RESETTOENTIRECIRCLE
   [no parameters]
```

Description: Reset the Theta-R Axis to inital settings. For Polar plots only.

Example:

```
$!VIEW RESETTOENTIRECIRCLE
```

$!VIEW SETMAGNIFICATION

Syntax:

```
$!VIEW SETMAGNIFICATION
   MAG = <dexp>
```

Description: Set the magnification for the data being plotted. A magnification of 1 will size the plot so it can fit within the grid area.

Required Parameter:

Parameters Syntax	Notes
`MAGNIFICATION = <dexp>`	

Example: Make the plot to be drawn one-half as big as when it fits within the grid area:

```
$!VIEW SETMAGNIFICATION
   MAGNIFICATION = 0.5
```

$!VIEW TRANSLATE

Syntax:

```
$!VIEW TRANSLATE
   X = <dexp>
   Y = <dexp>
   [no optional parameters]
```

Description: Shift the data being plotted in the X- and/or Y-direction. The amount translated is in frame units.

Required Parameters

Parameters Syntax	Default	Notes
`X = ` *<dexp>*	`0.0`	Amount to translate in X-frame units.
`Y = ` *<dexp>*	`0.0`	Amount to translate in Y-frame units.

Example: Translate the view 10 percent of the frame width to the right:

```
$!VIEW TRANSLATE
  X = 10
```

$!VIEW ZOOM

Syntax:

```
$!VIEW ZOOM
   X1 = <dexp>
   Y1 = <dexp>
   X2 = <dexp>
   Y2 = <dexp>
   [no optional parameters]
```

Description: Change the view by “zooming” into the data. In Sketch, XY, and 2D frame mode plots, Tecplot will adjust the ranges on the axis to view the region defined by the rectangle with corners at (X1, Y1) and (X2, Y2). For 3-D orthographic plots, the view is translated and scaled to fit the region. For 3-D perspective plots, the view is rotated about the viewer and scaled to fit the region. X1 and so forth are measured in grid coordinates.

Required Parameters:

Parameters Syntax	Notes
`X1 = ` *<dexp>*	
`Y1 = ` *<dexp>*	
`X2 = ` *<dexp>*	
`Y2 = ` *<dexp>*	

Example: Zoom so the rectangular region with corners at (1, 0) and (7, 9) are in view:

```
$!VIEW ZOOM
   X1 = 1
   Y1 = 0
```

```
X2 = 7
Y2 = 9
```

$!WHILE...$!ENDWHILE

Syntax: `$!WHILE` *<conditionalexp>*

```
   .
   .
   .
$!ENDWHILE
```

Description: Continue to execute a set of commands until a conditional expression is false.

Example: Execute a set of commands until the macro variable |**myvar**| is greater than 1.0:

```
$!VARSET |myvar| = 0.0
$!WHILE |myvar| < 1.0
   .
   .
   .
$!VARSET |myvar| + = 0.01
$!ENDWHILE
```

$!WORKSPACEVIEW *[Required-Control Option]*

Description: The different commands in the `WORKSPACEVIEW` compound function family are described separately in the following sections.

The `WORKSPACEVIEW` compound functions are:

```
$!WORKSPACEVIEW FITALLFRAMES
   $!WORKSPACEVIEW FITPAPER
   $!WORKSPACEVIEW FITSELECTEDFRAMES
   $!WORKSPACEVIEW LASTVIEW
   $!WORKSPACEVIEW MAXIMIZE
   $!WORKSPACEVIEW TRANSLATE
   $!WORKSPACEVIEW UNMAXIMIZE
   $!WORKSPACEVIEW ZOOM
```

$!WORKSPACEVIEW FITALLFRAMES

Syntax: `$!WORKSPACEVIEW FITALLFRAMES`
[no parameters]

Description: Change the view in the workspace so all frames are fit just inside the edges of the workspace.

Example: `$!WORKSPACEVIEW FITALLFRAMES`

$!WORKSPACEVIEW FITPAPER

Syntax: `$!WORKSPACEVIEW FITPAPER`
[no parameters]

Description: Change the view in the workspace so the entire paper is fit just inside the edges of the workspace.

Example: `$!WORKSPACEVIEW FITPAPER`

$!WORKSPACEVIEW FITSELECTEDFRAMES

Syntax: `$!WORKSPACEVIEW FITSELECTEDFRAMES`
[no parameters]

Description: Change the view in the workspace so the currently selected frames (that is, the frames with pick handles) are fit just inside the edges of the workspace.

Example: `$!WORKSPACEVIEW FITSELECTEDFRAMES`

$!WORKSPACEVIEW LASTVIEW

Syntax: `$!WORKSPACEVIEW LASTVIEW`
[no parameters]

Description: Return to the previous workspace view.

Example: `$!WORKSPACEVIEW LASTVIEW`

$!WORKSPACEVIEW MAXIMIZE

Syntax:

```
$!WORKSPACEVIEW MAXIMIZE
```

[no parameters]

Description: Temporarily expand the work area as large as possible. The maximized work area occupies the entire Tecplot process window.

Example:

```
$!WORKSPACEVIEW MAXIMIZE
```

$!WORKSPACEVIEW TRANSLATE

Syntax:

```
$!WORKSPACEVIEW TRANSLATE
   X = <dexp>
   Y = <dexp>
```

[no optional parameters]

Description: Shift the view of the workspace. This has no effect on the local view within any frame in your layout.

Required Parameters:

Parameters Syntax	Default	Notes
X = <dexp>	0	Value is in inches.
Y = <dexp>	0	Value is in inches.

Example: Shift the workspace view to the left by 2 inches (as measured by the workspace ruler):

```
$!WORKSPACEVIEW TRANSLATE
   X = -2
   Y = 0
```

$!WORKSPACEVIEW UNMAXIMIZE

Syntax:

```
$!WORKSPACEVIEW UNMAXIMIZE
```

[no parameters]

Description: Returns the workspace to its normal size after it has been expanded after **$!WORKSPACE MAXIMIZE** has been used.

Example: **$!WORKSPACEVIEW UNMAXIMIZE**

$!WORKSPACEVIEW ZOOM

Syntax:

```
$!WORKSPACEVIEW ZOOM
   X1 = <dexp>
   Y1 = <dexp>
   X2 = <dexp>
   Y2 = <dexp>
   [no optional parameters]
```

Description: Change the view into the work area. This has no effect on the local view within any frame in your layout.

Required Parameters:

Parameters Syntax	Notes
X1 = *<dexp>*	
Y1 = *<dexp>*	
X2 = *<dexp>*	
Y2 = *<dexp>*	

Example: Make the region in the lower left corner of an 8.5 by 11 paper be viewable in the work area. The paper is in portrait orientation:

```
$!WORKSPACEVIEW ZOOM
   X1 = 0
   Y1 = 5.5
   X2 = 4.25
   Y2 = 9.75
```

$!WRITECOLORMAP

Syntax:

```
$!WRITECOLORMAP <string>
   [no parameters]
```

Description: Write the current color map to a file. The <*string*> is the name of the file to write to.

Example:

```
$!WRITECOLORMAP "mycolors.map"
```

$!WRITECURVEINFO

Syntax:

```
$!WRITECURVEINFO <string>
   SOURCEMAP = <integer>
   [optional parameters]
```

Description: Write out the curve details or the calculated data points for the equation(s) used to draw the curve for a selected line mapping. The <*string*> is the name of the file to write to.

Required Parameter:

Parameter Syntax	Notes
`SOURCEMAP = <integer>`	This must be the number of an line mapping that does some type of curve fit or spline.

Optional Parameter:

Parameters Syntax	Default	Notes
`CURVEINFOMODE =` <*curveinfomode*>	`CURVE DETAILS`	Use `CURVE DETAILS` or `CURVEPOINTS.`

Example: Write out the coefficients for XY line mapping number 3 to `map3.out`:

```
$!WRITECURVEINFO "map3.out"
   SOURCEMAP      = 3
   CURVEINFOMODE  = CURVE DETAILS
```

$!WRITEDATASET

Syntax:

```
$!WRITEDATASET <string>
   [optional parameters]
```

Description: Write the data set attached to the current frame to a file. The <*string*> is the name of the file to write to.

Optional Parameters:

Parameters Syntax	Default	Notes
INCLUDETEXT = *<boolean>*	**TRUE**	
INCLUDEGEOM = *<boolean>*	**TRUE**	
INCLUDECUSTOMLABELS = *<boolean>*	**TRUE**	
INCLUDEDATA = *<boolean>*	**TRUE**	
INCLUDEDATASHARELINKAGE	**FALSE**	
INCLUDEAUTOGENFACENEIGHBORS	**FALSE**	
ASSOCIATELAYOUTWITHDATAFILE	**TRUE**	
VARPOSITIONLIST = *<set>*	All vars.	Use this to limit the number of variables written out.
ZONELIST = *<set>*	All zones.	Use this to limit the number of zones written out.
BINARY = *<boolean>*	**TRUE**	If **FALSE**, you can include **PRECISION** and **USEPOINTFORMAT**.
PRECISION = *<integer>*	**12**	Only used if ASCII (that is, **BINARY** is **FALSE**).
USEPOINTFORMAT = *<boolean>*	**FALSE**	Only used if ASCII (that is, **BINARY** is **FALSE**).

Example: Write out only zone 3 to a file called **zone3.plt**:

```
$!WRITEDATASET "zone3.plt"
   INCLUDETEXT          = FALSE
   INCLUDEGEOM          = FALSE
   INCLUDECUSTOMLABELS  = FALSE
   ZONELIST             = [3]
```

$!WRITESTYLESHEET

Syntax: **$!WRITESTYLESHEET** *<string>*
[optional parameters]

Description: Write the style for the current frame to a file. The <*string*> is the name of the file to write to.

Optional Parameters:

Parameters Syntax	Default	Notes
INCLUDECONTOURLEVELS = *<boolean>*	**TRUE**	
INCLUDETEXT = *<boolean>*	**TRUE**	
INCLUDEGEOM = *<boolean>*	**TRUE**	
INCLUDEPLOTSTYLE = *<boolean>*	**TRUE**	
INCLUDESTREAMPOSITIONS = *<boolean>*	**TRUE**	
INCLUDEFACTORYDEFAULTS = *<boolean>*	**FALSE**	
USERELATIVEPATHS = *<boolean>*		
INCLUDEAUXDATA = *<boolean>*	**TRUE**	

Example: Write out a stylesheet for the current frame to **f1.sty**:

```
$!WRITESTYLESHEET "f1.sty"
   INCLUDEFACTORYDEFAULTS = TRUE
```

$!XYLINEAXIS

Syntax:

```
$!XYLINEAXIS
```

[optional parameters]

Description: A SetValue command that assigns attributes for axes in an XY Line plot.

Optional Parameters:

Parameter Syntax		Notes
DEPXTOYRATIO	*<op> <dexp>*	**AXISMODE** must be **XYDEPENDENT** to use this. This applies only to the X1- and Y1-axes.
AXISMODE	**=** *<axismode>*	Set to **INDEPENDENT** or **XYDEPENDENT**.
GRIDAREA	*<<gridarea>>*	
XDETAIL	*<integer> <<axisdetail>>*	The *<integer>* option specifies which axis to operate on, 1 £ n £ 5.

Parameter Syntax		Notes
YDETAIL	*<integer>* *<<axisdetail>>*	The *<integer>* option specifies which axis to operate on, 1 £ *n* £ 5.
PRECISEGRID	*<<precisegrid>>*	
VIEWPORTTOPSNAPTARGET	**=** *<integer>*	Default = 100
VIEWPORTTOPSNAPTOLERANCE	**=** *<integer>*	Default = 10
VIEWPORTNICEFITBUFFER	**=** *<double>*	Between 1 and 100.
AUTOADJUSTRANGESTONICEVALUES	**=** *<boolean>*	
PRESERVEAXISSCALE	**=** *<boolean>*	

Example: Set the axis mode to be independent for the XY-axes (note that this affects only X1 versus Y1):

```
$!XYLINEAXIS
   AXISMODE = INDEPENDENT
```

CHAPTER 6 Parameter Subcommands

This chapter details secondary or common macro parameter subcommands in Tecplot. These subcommands provide a means to access the lower level variables of commands defined in the previous chapter of this manual. Each subcommand can expand to contain one or more parameters or subcommands. All parameters within a subcommand are optional.

Items within single angle brackets (< >) are defined in Chapter 7, “Parameter Assignment Values, Expressions, and Arithmetic and Logical Operators.”

<<anchorpos>>

Description: Assign attributes for positioning of objects.

Expands to:

Syntax		Notes
{		
X	= *<double>*	Sets X-value (and THETA-value)
Y	= *<double>*	Sets Y-value (and R-value)
Z	= *<double>*	Sets Z-value
THETA	= *<double>*	Sets THETA-value (and X-value)
R	= *<double>*	Sets R-value (and Y-value)
}		

Example: Make a square geometry and place it at a certain XY location:

```
$!ATTACHGEOM
  GEOMTYPE = SQUARE
  POSITIONCOORDSYS = FRAME
```

```
    ANCHORPOS
      {
      X = 2.89124668435
      Y = 88.7359084881
      }
    RAWDATA
  5.23430593312
```

<<areastyle>>

Description: Change settings for the axis grid area.

Expands to:

Syntax		Notes
`{`		
`DRAWGRIDLAST`	= *<boolean>*	Not available in 3D frame mode.
`DRAWBORDER`	= *<boolean>*	
`LINETHICKNESS`	*<op> <dexp>*	
`COLOR`	= *<color>*	
`ISFILLED`	= *<boolean>*	
`FILLCOLOR`	= *<color>*	
`USELIGHTSOURCETOFILL`	= *<boolean>*	Only available for 3D frame mode.
`}`		

Example: Turn on the grid area border for a 2-D plot and change the line thickness to be 2 percent:

```
$!TWODAXIS
   AREASTYLE
   {
    DRAWBORDER = YES
    LINETHICKNESS = 2
   }
```

<<axisdetail>>

Description: Assign attributes for axes.

Expands to:

Syntax	Notes

{
SHOWAXIS = *<boolean>*
AUTOGRID = *<boolean>*
ISREVERSED = *<boolean>*
GRANCHOR = *<double>*
GRSPACING = *<double>*
RANGEMIN = *<double>*
RANGEMAX = *<double>*
COORDSCALE = *<coordscale>*
CLIPDATA = *<boolean>*
VALUEATORIGIN = *<double>*
VARNUM = *<integer>*
TICKLABEL *<<ticklabeldetail>>*
GRIDLINES *<<gridlinedetails>>*
MINORGRIDLINES *<<gridlinedetails>>*
TICKS *<<tickmarkdetail>>*
TITLE *<<axistitle>>*
AXISLINE *<<axisline>>*
}	

Example: Turn on the axis line, reverse the axis direction, and set the range to go from 0.5 to 1.5 for the X-axis in a 2-D plot:

```
$!TWODAXIS
   SHOWAXISLINE = TRUE
   XDETAIL
   {
    ISREVERSED = TRUE
    RANGEMIN   = 0.5
    RANGEMAX   = 1.5
   }
```

<<axisline>>

Description: Assign attributes for axis lines.

Expands to:

Syntax		Notes
{		
SHOW	= *<boolean>*	
SHOWBOTHDIRECTIONS	= *<boolean>*	Non-3D only. Default = FALSE
SHOWPERPENDICULAR	= *<boolean>*	Non-3D only. Default = **FALSE**
SHOWOPPOSITEEDGE	= *<boolean>*	3D only. Default = **FALSE**
COLOR	= *<color>*	
LINETHICKNESS	= *<double>*	
ALIGNMENT	= *<axisalignment>*	
OPPOSINGAXISVALUE	= *<double>*	
POSITION	= *<double>*	
ANGLE	= *<double>*	
OFFSET	= *<double>*	
EDGE	= *<integer>*	
}		

Example: Change the thickness of the Theta-axis line to 0.8 and the color to red.:

```
$!POLARAXIS THETADETAIL{AXISLINE{COLOR = RED}}
$!POLARAXIS THETADETAIL{AXISLINE{LINETHICKNESS = 0.8}}
```

<<axistitle>>

Description: Assign attributes for titles.

Expands to:

Syntax	Notes
{	
SHOWONAXISLINE = *<boolean>*	Default = **TRUE**
SHOWONGRIDBORDERMIN = *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONGRIDBORDERMAX = *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONOPPOSITEEDGE = *<boolean>*	3D only. Default = **FALSE**
SHOWONALLAXES = *<boolean>*	Polar R only. Default = **TRUE**
SHOWONVIEWPORTTOP = *<boolean>*	Polar only. Default = **TRUE**
SHOWONVIEWPORTBOTTOM = *<boolean>*	Polar only. Default = **TRUE**
SHOWONVIEWPORTLEFT = *<boolean>*	Polar only. Default = **TRUE**
SHOWONVIEWPORTRIGHT = *<boolean>*	Polar only. Default = **TRUE**
TITLEMODE = *<titlemode>*	
TEXT = *<string>*	
COLOR = *<color>*	
TEXTSHAPE = *<<textshape>>*	
OFFSET = *<double>*	
PERCENTALONGLINE = *<double>*	Default = 50%
}	

Example: Create a R-axis title, saying “Harmonic Motion” in red, times, size 6 font.:

```
$!POLARAXIS RDETAIL{TITLE{TEXT = ’Harmonic Motion’}}
$!POLARAXIS RDETAIL{TITLE{OFFSET = -4}}
$!POLARAXIS RDETAIL{TITLE{COLOR = RED}}
$!POLARAXIS RDETAIL{TITLE{TEXTSHAPE{FONT = TIMES}}}
$!POLARAXIS RDETAIL{TITLE{TEXTSHAPE{HEIGHT = 6}}}
```

<<basicsizelist>>

Description: Assign basic sizes. The units for the values assigned here are dependent on the parent command. Assignments here do not affect the plot. These assignments are used only to configure drop-down menus in the interface so the user can make quick selections.

Expands to:

Syntax	Notes
{	
TINY *<op> <dexp>*	
SMALL *<op> <dexp>*	
MEDIUM *<op> <dexp>*	
LARGE *<op> <dexp>*	
HUGE *<op> <dexp>*	
}	

Example: Change the medium line pattern length for drop-down menus in the interface to be five percent:

```
$!BASICSIZE
   LINEPATLENGTHS
   {
    MEDIUM = 5
   }
```

<<colormapcontrolpoints>>

Description: All contour color maps except the Raw user-defined color map make use of control points to determine the color distribution. Each control point has a position and a left and right color. The *<<colormapcontrolpoints>>* subcommand can contain more than one **CONTROLPOINT** subcommand.

Expands to:

Syntax	Notes
{	
CONTROLPOINT *<integer>*	Use *<integer>* to specify which control point to modify.
{	
COLORMAPFRACTION *<op> <dexp>*	Positions the control point; 0 sets the position to the lowest index and 1 to the highest index in the color map.
LEADRGB *<<rgb>>*	
TRAILRGB *<<rgb>>*	
}	
}	

Example: Change the lead RGB values for control point 2 in the small rainbow color map to be 100, 0, 0:

```
$!COLORMAP
   SMRAINBOW
   {
    CONTROLPOINT 2
    {
     LEADRGB
     {
      R = 100
      G =   0
      B =   0
     }
    }
   }
```

<<colormapoverride>>

Description: Change settings for a color map override. Color map overrides are used to replace a specific band in a contour color map with one of the 16 basic colors.

Expands to:

Syntax	Notes

{
INCLUDE = *<boolean>*
COLOR = *<color>*
STARTLEVEL *<op> <integer>*
ENDLEVEL *<op> <integer>*
}	

Example: Set the color used between contour level number 1 to number 3 to be purple. Use color map override number 3:

```
$!GLOBALCONTOUR
   COLORMAPFILTER
   {
    COLORMAPOVERRIDEACTIVE = YES
    COLORMAPOVERRIDE 3
    {
     INCLUDE = YES
     COLOR = PURPLE
     STARTLEVEL = 1
     ENDLEVEL   = 3
    }
  }
```

<<continuouscolor>>

Description: Change settings for continuous color.

Expands to:

Syntax		Notes
`CMIN` `CMAX`	= *<boolean>* = *<boolean>*	

Example: Set the continuous color.

```
$!GLOBALCONTOUR VAR = 4
$!FIELDLAYERS SHOWCONTOUR = YES

$!GLOBALCONTOUR COLORMAPFILTER
  {COLORMAPDISTRIBUTION = CONTINUOUS}
$!GLOBALCONTOUR COLORMAPFILTER
  {
   CONTINUOUSCOLOR
    {
     CMIN = 0.5
     CMAX = 2
    }
  }
```

<<gridlinedetail>>

Description: Change settings for axis gridlines.

Expands to:

Syntax	Notes
{	
SHOW = *<boolean>*	
LINEPATTERN = *<linepattern>*	
PATTERNLENGTH *<op> <dexp>*	
LINETHICKNESS *<op> <dexp>*	
CUTTOFF = *<double>*	Theta only.
}	

Example: Set the line pattern for minor gridlines for the X-axis in a 3-D plot to be dashed:

```
$!THREEDAXIS
   XDETAIL
   {
    MINORGRIDLINES
    {
     LINEPATTERN = DASHED
    }
   }
```

<<ijk>>

Description: Set an I-, J- or K-index.

Expands to:

Syntax	Notes
{	
I *<op> <integer>*	
J *<op> <integer>*	
K *<op> <integer>*	
}	

Example: Set the I- and J-index skip for vectors to 2 for all zones:

```
$!FIELD
   VECTOR
   {
    IJKSKIP
    {
```

```
    I = 2
    J = 2
  }
}
```

<<indexrange>>

Description: Set an index range.

Expands to:

Syntax	Notes

`{`
`MIN` *<op> <integer>*
`MAX` *<op> <integer>*
`SKIP` *<op> <integer>*
`}`	

Example: Change the plot so the data set shows I-planes 3, 5, and 7 for zones 1 to 3:

```
$!FIELD [1-3]
   SURFACES
   {
    SURFACESTOPLOT = IPLANES
    IRANGE
    {
      MIN  = 3
      MAX  = 7
      SKIP = 2
    }
   }
```

<<dialogplacement>>

Description: Describes the placement for a dialog.

Expands to:

Syntax	Notes

`{`
`ANCHORALIGNMENT = <anchoralignment>`
`ANCHORHORIZONTALINSIDE = <boolean>`
`ANCHORVERTICALINSIDE = <boolean>`
`MINVISIBILITYPERCENTAGE = <integer>`
`XOFFSET = <integer>`
`YOFFSET = <integer>`
`PLACEATANCHOR = <placeatanchor>`
`}`
	XOFFSET and YOFFSET are in pixels. They may be negative, but will be truncated to the bounding rectangle of the Tecplot main window.
	ANCHORHORIZONTALINSIDE and ANCHORVERTICALINSIDE control how the dialog window is anchored in both the horizontal and vertical directions relative to the Tecplot main window. The MINVISIBILITYPERCENTAGE specifies the minimum percentage of the dialog, between 1 and 100, that must be visible within the desktop. This prevents a dialog from being placed outside of the visible desktop. Note that not all window managers allow dialogs to be placed so that the portions of the dialog are not visible and in effect enforce a value of 100.
	PLACEATANCHOR specifies when to place it at the anchor, NEVER, ONCE (initial launch), or ALWAYS.

Example: Set the position of the Colormap dialog to always launch 10 pixels from Tecplot's bottom-right corner:

```
$!INTERFACE
   DIALOGPLACEMENT
   {
    COLORMAPDIALOG
    {
     ANCHORALIGNMENT = BOTTOMRIGHT
     XOFFSET = 10
     YOFFSET = 10
     PLACEATANCHOR = ALWAYS
    }
   }
```

<<numberformat>>

Description: Set the format used to draw a number.

Expands to:

Syntax		Notes
{		
FORMATTING	= *<valueformat>*	
CUSTOMLABEL	= *<integer>*	
PRECISION	*<op> <integer>*	
SHOWDECIMALSONWHOLENUMBERS	= *<boolean>*	Default = **FALSE**
REMOVELEADINGZEROS	= *<boolean>*	Default = **FALSE**
SHOWNEGATIVESIGN	= *<boolean>*	Default = **TRUE**
POSITIVEPREFIX	= *<string>*	
POSITIVESUFFIX	= *<string>*	
NEGATIVEPREFIX	= *<string>*	
NEGATIVESUFFIX	= *<string>*	
ZEROPREFIX	= *<string>*	
ZEROSUFFIX	= *<string>*	
}		

Example: Set the number format for axis labels on the X-axis in a 2-D field plot to use the “float” format with a precision of 3, and add the phrase “DAYS WITHOUT RAIN” after every positive value:

```
$!TWODAXIS
   XDETAIL
   {
    TICKLABEL
    {
     NUMFORMAT
     {
      FORMATTING = FIXEDFLOAT
       PRECISION = 3
      POSITIVESUFFIX = “DAYS WITHOUT RAIN”
     }
    }
   }
```

<<papersize>>

Description: Change dimensions or hardclip offsets for **LETTER, DOUBLE, A3, A4,**

CUSTOM1 and **CUSTOM2** paper sizes.

Expands to:

Syntax	Notes

`{`
`WIDTH` *<op> <dexp>*
`HEIGHT` *<op> <dexp>*
`LEFTHARDCLIPOFFSET` *<op> <dexp>*
`RIGHTHARDCLIPOFFSET` *<op> <dexp>*
`TOPHARDCLIPOFFSET` *<op> <dexp>*
`BOTTOMHARDCLIPOFFSET` *<op> <dexp>*
`}`
	All values are in inches.

Example: Change the left hardclip offset for **LETTER** size paper to be 0.25 inches:

```
$!PAPER
   PAPERSIZEINFO
   {
    LETTER
    {
     LEFTHARDCLIPOFFSET = 0.25
    }
   }
```

<<plotterpenmap>>

Description: Assign plotter pens to objects or colors for hardcopy output to pen plotters. Some objects are assigned a pen regardless of their color. All other objects are assigned a pen based on their color.

Expands to:

Syntax	Notes

{
BLACKPEN = *<integer>*
REDPEN = *<integer>*
GREENPEN = *<integer>*
BLUEPEN = *<integer>*
CYANPEN = *<integer>*
YELLOWPEN = *<integer>*
PURPLEPEN = *<integer>*
WHITEPEN = *<integer>*
CUSTOM1PEN = *<integer>*
CUSTOM2PEN = *<integer>*
CUSTOM3PEN = *<integer>*
CUSTOM4PEN = *<integer>*
CUSTOM5PEN = *<integer>*
CUSTOM6PEN = *<integer>*
CUSTOM7PEN = *<integer>*
CUSTOM8PEN = *<integer>*
AXISPEN = *<integer>*
MAJGRIDLINEPEN = *<integer>*
MINGRIDLINEPEN = *<integer>*
STREAMLINEPEN = *<integer>*
MULTICOLORLINEPEN = *<integer>*
BOUNDARYPEN = *<integer>*
LABELPEN = *<integer>*
}
	Factory default for all objects is to use pen1.

Example: Make the drawing of all axes use pen 3:

```
$!PRINTSETUP
   PLOTTERPENMAP
   {
    AXISPEN = 3
   }
```

<<precisegrid>>

Description: Change settings for the precise dot grid.

Expands to:

Syntax	Notes

{
INCLUDE = *<boolean>*
COLOR = *<color>*
SIZE = *<double>*
}
	Size is in centimeters.

Example: Turn on the precise dot grid in an XY-plot:

```
$!XYAXIS
   PRECISEGRID
     {
      INCLUDE = YES
     }
```

<<rect>>

Description: Change settings for a rectangle. The rectangle is defined using two points (X1,Y1) and (X2,Y2).

Expands to:

Syntax	Notes

{
X1 *<op> <dexp>*
Y1 *<op> <dexp>*
X2 *<op> <dexp>*
Y2 *<op> <dexp>*
}
	Units are based on the parent command.

Example: Set the 2-D axis grid area to be positioned 10 percent from all edges of the frame:

```
$!TWODAXIS
   AREASTYLE
   {
    EXTENTS
    {
     X1 = 10
     Y1 = 10
     X2 = 90
     Y2 = 90
```

```
         }
        }
```

<<refscatsymbol>>

Description: Set the attributes for the reference scatter symbol.

Expands to:

Syntax	Notes

{
SHOW = *<boolean>*
COLOR = *<color>*
LINETHICKNESS = *<dexp>*
ISFILLED = *<boolean>*
FILLCOLOR = *<color>*
MAGNITUDE = *<dexp>*
XYPOS *<<xy>>*
SYMBOLSHAPE *<<symbolshape>>*
}	

Example: Change the fill color of the reference scatter symbol to be green:

```
$!GLOBALSCATTER
   REFSCATSYMBOL
   {
    FILLCOLOR = GREEN
   }
```

<<renderconfig>>

Description: Set the attributes for OpenGL rendering.

Expands to:

Syntax	Notes

{
POLYGONOFFSETEXTBIASFACTOR = *<double>*
STIPPLEALLLINES = *<stipplemode>*
DEPTHBUFFERSIZE = *<integer>*
MINBITSPERRGBPLANE = *<integer>*
DOEXTRADRAWFORLASTPIXEL = *<boolean>*
MAXSTRIPLENGTH = <integer>
MAXPRIMATIVESPERBLOCK = <integer>
CONSTANTLYUSESCISSORING = *<boolean>*
USEQUADSTRIPS = *<boolean>*
USETRIANGLESTRIPS = *<boolean>*
TRIANGULATEFILLEDPOLYGONS = *<boolean>*
USEGLCOLORMATERIALFUNCTION = *<boolean>*
MAXTEXTURESIZE = *<integer>*
FORCESMOOTHSHADINGFORLIGHTING = *<boolean>*
ADJUSTRECTANGLERIGHTANDBOTTOM = *<boolean>*
}
	If thin patterned lines are not drawn correctly, set **STIPPLEALLLINES** to **ALL**.
	For low memory graphics cards, the depth buffer size may need to be reduced.
	Specify the minimum number of bits used for each of the planes in the image buffer.
	Sometimes the last pixel for stroked font characters is not drawn If so, turn **DOEXTRADRAWFORLASTPIXEL** on.
	Some graphics cards have problems with long strips. Use **MAXSTRIPLENGTH** to reduce the strip length.
	Some graphics cards have problems with large numbers of graphics primitives in a single block. Use MAXPRIMATIVESPERBLOCK to reduce the number of primitives delivered to the graphics hardware in a single block.
	Turn **ConstantlyUseScissoring** on if you see lines extending outside the borders of the frame. There is a slight performance penalty when using this option.
	If some shaded or contour flooded quads or triangles do not appear or are black, try turning this off.
	As with **USEQUADSTRIPS**, try turning off **USEQUADSTRIPS** before turning **USETRIANGLESTIPS** off. Turning off both options will result in reduced performance, but may help fix errors caused by buggy graphics card drivers.
	As with **USEQUADSTRIPS**, try turning on **TRIANGULATEFILLEDPOLYGONS** if you are still experiencing problems even after turning off **USETRIANGLESTRIPS** and **USEQUADSTRIPS**.
	Some graphics cards have problems with an OpenGL's **glColorMaterial** function. Higher performance (especially for continuous contour flooded plots) can be achieved when it is used. However, it may need to be turned off if you are experiencing problems.

Example: Force all line drawing to include the last point in the line. Also, make the size of the depth buffer to be at least 32 bits.

```
$!INTERFACE
   OPENGLCONFIG
   {
     SCREENRENDERING
      {
        DOEXTRADRAWFORLASTPIXEL = TRUE
        DEPTHBUFFERSIZE = 32
      }
   }
```

<<*rgb*>>

Description: Set a color value by assigning values to its red, green, and blue components.

Expands to:

Syntax	Notes

{
R *<op> <integer>*
G *<op> <integer>*
B *<op> <integer>*
}	

Example: Change the **CUSTOM3** basic color to be light green:

```
$!BASICCOLOR
   CUSTOM 3
   {
    R =  80
    G = 255
    B =  80
   }
```

<<*shademap*>>

Description: Map colors on the screen to shades of gray for monochrome hardcopy output.

Expands to:

Syntax	Notes

{
BLACKSHADE = <*dexp*>
REDSHADE = <*dexp*>
GREENSHADE = <*dexp*>
BLUESHADE = <*dexp*>
CYANSHADE = <*dexp*>
YELLOWSHADE = <*dexp*>
PURPLESHADE = <*dexp*>
WHITESHADE = <*dexp*>
CUSTOM1SHADE = <*dexp*>
CUSTOM2SHADE = <*dexp*>
CUSTOM3SHADE = <*dexp*>
CUSTOM4SHADE = <*dexp*>
CUSTOM5SHADE = <*dexp*>
CUSTOM6SHADE = <*dexp*>
CUSTOM7SHADE = <*dexp*>
CUSTOM8SHADE = <*dexp*>
}
	Shade values can range from 0 (black) to 100 (white).

Example: Make blue flooded regions map to 50 percent gray:

```
$!PRINTSETUP
   MONOFLOODMAP
   {
    BLUESHADE = 50
   }
```

<<symbolshape>>

Description: Set a symbol shape. Symbols can be a geometric shape (circle, square, and so forth) or an ASCII character.

Expands to:

Syntax	Notes

{
ISASCII = *<boolean>*
ASCIISHAPE = *<string>*
{ = *<geomshape>*
USEBASEFONT = *<boolean>*
FONTOVERRIDE = *<font>*
CHAR = *<string>*
}
GEOMSHAPE = *<geomshape>*
}	

Example: Change the symbol shape for symbols drawn with line map 3 to use circles:

```
$!LINEMAP[3]
   SYMBOLS
   {
    SYMBOLSHAPE
    {
     ISASCII = FALSE
     GEOMSHAPE = CIRCLE
    }
   }
```

<<textbox>>

Description: Change settings for the optional box around a text label.

Expands to:

Syntax	Notes

{
BOXTYPE = *<textboxtype>*
MARGIN *<op> <dexp>*
LINETHICKNESS *<op> <dexp>*
COLOR = *<color>*
FILLCOLOR = *<color>*
}	

Example: See example for *<<textshape>>*.

<<textshape>>

Description: Change settings related to text font and character height.

Expands to:

Syntax	Notes

`{`
`FONT` = *<font>*
`SIZEUNITS` = *<sizeunits>*
`HEIGHT` *<op> <dexp>*
`}`	

Example: Add a text label in the center of the frame using Times Roman font. Make the text height 12 point. Include a box around the text with a line thickness of one percent:

```
$!ATTACHTEXT
   XYPOS {
    X = 50
    Y = 50
   }
   TEXTSHAPE
   {
    FONT = TIMES
   }
   BOX
   {
    BOXTYPE = HOLLOW
    LINETHICKNESS = 1
   }
   TEXT = 'Hi Mom'
```

<<ticklabeldetail>>

Description: Change settings for the text used to label axis tick marks.

Expands to:

Syntax		Notes
{		
SHOWONAXISLINE	= *<boolean>*	Default = **TRUE**
SHOWONGRIDBORDERMIN	= *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONGRIDBORDERMAX	= *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONOPPOSITEEDGE	= *<boolean>*	3D only. Default = **FALSE**
SHOWONALLAXES	= *<boolean>*	Polar R only. Default = **TRUE**
SHOWATAXISINTERSECTION	= *<boolean>*	
SKIP	= *<integer>*	
ERASEBEHINDLABELS	= *<boolean>*	
NUMFORMAT	*<<numberformat>>*	
TEXTSHAPE	*<<textshape>>*	
OFFSET	*<op> <dexp>*	Not allowed to change size units parameter.
LABELALIGNMENT	= *<labelalignment>*	
ANGLE	*<op> <dexp>*	
COLOR	= *<color>*	
}		

Example: Change the color for X-axis tick mark labels in a 2-D plot to be red:

```
$!TWODAXIS
   XDETAIL
   {
    TICKLABEL
    {
     COLOR = RED
    }
   }
```

<<tickmarkdetail>>

Description: Assign attributes for axis tick marks.

Expands to:

Syntax	Notes
{	
SHOWONAXISLINE = *<boolean>*	Default = **TRUE**
SHOWONGRIDBORDERMIN = *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONGRIDBORDERMAX = *<boolean>*	Non-3D only. Default = **FALSE**
SHOWONOPPOSITEEDGE = *<boolean>*	3D only. Default = **FALSE**
SHOWONALLAXES = *<boolean>*	Polar R only. Default = **TRUE**
TICKDIRECTION = *<tickdirection>*	
LENGTH *<op> <dexp>*	
LINETHICKNESS *<op> <dexp>*	
NUMMINORTICKS = *<integer>*	
MINORLENGTH = *<double>*	
MINORLINETHICKNESS = *<double>*	
}	

Example: Set the tick mark length to 2 percent for the second Y-axis in an XY-plot:

```
$!XYLINEAXIS
   YDETAIL 2
   {
    TICKS
    {
     LENGTH = 2
      SHOWONGRIDBORDERMIN = TRUE
    }
   }
```

<<volumeobjectstoplot>>

Description: Specifies what volume objects are to be displayed.

Expands to:

Syntax	Notes
{	
SHOWISOSURFACES = *<boolean>*	
SHOWSLICES = *<boolean>*	
SHOWSTREAMTRACES = *<boolean>*	
}	

Example: **$!FIELD**

```
VOLUMEMODE
 {
  VOLUMEOBJECTSTOPLOT
   {
    SHOWISOSURFACES = NO
    SHOWSLICES = YES
    SHOWSTREAMTRACES = YES
   }
 }
```

<<xy>>

Description: Change settings for an (X,Y) position.

Expands to:

Syntax	Notes

{
X *<op> <dexp>*
Y *<op> <dexp>*
}	

Example: See the **XYPOS** parameter in the example for *<<textshape>>*.

<<xyz>>

Description: Change settings for an (X, Y, Z) triplet.

Expands to:

Syntax	Notes

{
X *<op> <dexp>*
Y *<op> <dexp>*
Z *<op> <dexp>*
}	

Example: Change the scale factor on the Z-axis to be 0.5:

```
$!GLOBALTHREED
   AXISSCALEFACT
   {
    Z = 0.5
   }
```

<<*zebrashade*>>

Description: Change zebra shading attributes.

Expands to:

Syntax	Notes

`{`
`INCLUDE` = <*boolean*>
`ISTRANSPARENT` = <*boolean*>
`COLOR` = <*color*>
`}`	

Example: Turn on zebra shading and make the zebra shade color to be black:

```
$!GLOBALCONTOUR
   COLORMAPFILTER
   {
    ZEBRA
    {
     INCLUDE = TRUE
     COLOR   = BLACK
    }
   }
```

CHAPTER 7 Parameter Assignment Values, Expressions, and Arithmetic and Logical Operators

7.1. Assignment Value Table

Parameter assignments referenced in the previous chapters using single angle brackets (< >) are defined here. (Case is not important.)

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<addonstyle>	`V7STANDARD,V7ACTIVEX`
<altmousebuttonmode>	`REDRAW, REVERTTOSELECT`
<anglespec>	`RADIANS, DEGREES`
<arrowheadattachment>	`NONE, ATBEGINNING, ATEND, ATBOTHENDS`
<arrowheadstyle>	`PLAIN, FILLED, HOLLOW`
<axisalignment>	`WITHVIEWPORT, WITHOPPOSINGAXISVALUE, WITHGRIDMIN, WITHGRIDMAX, WITHSPECIFICANGLE, WITHGRIDAREATOP, WITHGRIDAREABOTTOM, WITHGRIDAREALEFT, WITHGRIDAREARIGHT.`
<axismode>	`INDEPENDENT, XYDEPENDENT, XYZDEPENDENT`
<axistitlemode>	`USEVARNAME, USETEXT`
<axistitleposition>	`LEFT, CENTER, RIGHT`
<backingstoremode>	`NOTUSED, REALTIMEUPDATE, PERIODICUPDATE`
<bitdumpregion>	`CURRENTFRAME, ALLFRAMES, WORKAREA`
<boolean>	`YES, NO, TRUE, FALSE, ON, OFF`
<boundarycondition>	`FIXED, ZEROGRADIENT, ZERO2ND`
<boundarysetting>	`NONE, MIN, MAX, BOTH`
<boxtype>	`NONE, FILLED, HOLLOW`
<charactersequence>	One or more printable characters.

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<clipping>	CLIPTOVIEWPORT, CLIPTOFRAME
<color>	BLACK, RED, GREEN, BLUE, CYAN, YELLOW, PURPLE, WHITE, CUSTOM1 to CUSTOM56, MULTI1, MULTI2, MULTI3, MULTI4, RGBCOLOR
<colormap>	*<standardcolormap>*, WILD, USERDEF, RAWUSERDEF
<colormapcontrol>	COPYSTANDARD, REDISTRIBUTECONTROLPOINTS, RESETTOFACTORY
<colormapdistribution>	BANDED, CONTINUOUS
<conditionalexp>	*<dexp> <relop> <dexp>* or *<string> <relop> <string>*.
<contourcoloring>	RGB, GROUP1, GROUP2, GROUP3, GROUP4
<contourlabelaction>	ADD, DELETEALL
<contourlevelaction>	ADD, DELETENEAREST, DELETERANGE, NEW, RESET
<contourlinemode>	USEZONELINETYPE, SKIPTOSOLID, DASHNEGATIVE
<contourtype>	LINES, FLOOD, BOTHLINESANDFLOOD, AVERAGECELL, PRIMARYVALUE
<coordscale>	LINEAR, LOG
<coordsys>	GRID, FRAME, GRID3D
<curveinfomode>	CURVEDETAILS, CURVEPOINTS
<curvetype>	LINESEG, CURVFIT, SPLINE, PARASPLINE, ETORFIT, POWERFIT, EXTENDED
<datatype>	SINGLE, DOUBLE, LONGINT, SHORTINT, BYTE, BIT
<derivpos>	SIMPLE, ATPOINT, COMPLEX, ATPOINTB2
<dexp>	*<double>*, ((*<expression>*))
<double>	Valid floating point value.
<draworder>	BEFOREDATA, AFTERDATA
<drift>	NONE, LINEAR, QUAD
<epspreviewimagetype>	NONE, TIFF, EPSIV2, FRAME
<errorbartype>	UP, DOWN, LEFT, RIGHT, VERT, HORZ, CROSS
<exportformat>	RASTERMETAFILE, TIFF, SUNRASTER, XWINDOWS, PSIMAGE, HPGL, HPGL2, PS, EPS, WINDOWSMETAFILE, BMP, PNG, AVI, JPEG
<expression>	See Section 7.2.
<fillmode>	NONE, USESPECIFICCOLOR, USEBACKGROUNDCOLOR, USELINECOLOR

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<font>	HELV, HELVBOLD, TIMES, TIMESBOLD, TIMESITALIC, TIMESITALICBOLD, COURIER, COURIERBOLD, GREEK, MATH, USERDEF
<frameaction>	DELETETOP, FITALLTOPAPER, POP, POPATPOSITION, PUSHTOP
<framecollection>	ALL, PICKED
<framemode>	THREED, TWOD, XY, SKETCH
<functiondependency>	XINDEPENDENT, YINDEPENDENT, THETAINDEPENDENDT, RINDEPENDENT
<geomshape>	SQUARE, DEL, GRAD, RTRI, LTRI, DIAMOND, CIRCLE, CUBE, OCTAHEDRON, SPHERE, POINT
<geomtype>	GEOMIMAGE, LINESEGS, RECTANGLE, SQUARE, CIRCLE, ELLIPSE, LINESEGS3D
<ijkblankmode>	INTERIOR, EXTERIOR
<ijklines>	I, J, K
<ijkplane>	I, J, K
<imagestyle>	ONEPERFRAME, WORKSPACEONLY
<anchoralignment>	TOPLEFT, TOPCENTER, TOPRIGHT, MIDDLELEFT, MIDDLECENTER, MIDDLERIGHT, BOTTOMLEFT, BOTTOMCENTER, BOTTOMRIGHT
<integer>	Valid integer value.
<interpptselection>	ALLPOINTS, NEARESTNPOINTS, OCTANTNPOINTS
<isosurfacesselection>	ALLCOUNTOURLEVELS, ONESPECIFICVALUE, TWOSPECIFICVALUES, THREESPECIFICVALUES
<krigdrift>	NONE, LINEAR, QUAD
<labelalignment>	BYANGLE, ALONGAXIS, PERPENDICULARTOAXIS
<labeltype>	INDEX, VARVALUE, XANDYVARVALUE[a]
<lightingeffect>	PANELED, GOURAUD
<linearinterpmode>	DONTCHANGE, SETTOCONST
<linepattern>	SOLID, DASHED, DASHDOT, DOTTED, LONGDASH, DASHDOTDOT
<linktype>	WITHINFRAME, BETWEENFRAMES
<macrofunctionvar>	|*<integer>*|

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<macrointrinsic>	`IS3DV, LOOP, NUMVARS, NUMFRAMES, NUMZONES, OPSYS, NUMPLANES, TECHOME, MINB, MAXB, MINC, MAXC, MINS, MAXS, MINU, MAXU, MINV, MAXV, MINW, MAXW, MINX, MAXX, MINY, MAXY, MINZ, MAXZ, MAXI, MAXJ, MAXK, NUMWIN, NUMXYMAPS, COLORMAPDYNAMIC, TECPLOTVERSION, MINVnn, MAXVnn, AXISMINX, AXISMAXX, AXISMINY, AXISMAXY, AXISMINZ, AXISMAXZ, STARTSLICEPOS, ENDSLICEPOS, SLICEPLANETYPE, MACROFILEPATH, PLATFORM, FRAMEMODE`
<macrointrinsicvar>	|*<macrointrinsic>*|
<macroparameter>	*<charactersequence>, <string>*
<macroparameterlist>	`(,` *<macroparameter>, <macroparameter>*`, ...)`
<macrouserdefvar>	|*<charactersequence>*|
<macrovar>	*<macrointrinsicvar>, <macrouserdefvar>, <macrofunctionvar>*
<meshtype>	`WIREFRAME, OVERLAY, HIDDENLINE`
<mirrorvar>	`‘X’, ‘Y’, ‘Z’`
<mousebuttonclick>	`REDRAW, REVERTTOSELECT, NOOP`
<mousebuttondrag>	`NOOP, ZOOMDATA, ZOOMPAPER, TTRANSLATEDATA, TRANSLATEPAPER, ROLLERBALLROTATE, SPHERICALROTATE, XROTATE, YROTATE, ZROTATE, TWISTROTATE`
<mousemode>	`ADJUST, SELECT`
<noncurrentframedrawlevel>	`FULL, TRACE`
<objectalign>	`BOTTOM, CENTER, TOP, LEFTJUSTIFY, RIGHTJUSTIFY`
<op>	`=, -=, +=, *=, /=`
<originresetlocation>	`DATACENTER, VIEWCENTER`
<palette>	`MONOCHROME, PENPLOTTER, COLOR`
<papergridspacing>	`HALFCENTIMETER, ONECENTIMETER, TWOCENTIMETERS, QUARTERINCH, HALFINCH, ONEINCH, TENPOINTS, TWENTYFOURPOINTS, THIRTYSIXPOINTS, FIFTYPOINTS`
<paperrulerspacing>	`ONECENTIMETER, TWOCENTIMETERS, ONEINCH, FIFTYPOINTS, SEVENTYTWOPOINTS`
<papersize>	`LETTER, DOUBLE, A4, A3, CUSTOM1, CUSTOM2`
<pickaction>	`ADD, ADDALL, ADDALLINREGION, CLEAR, COPY, CUT, EDIT, MAGNIFY, PASTE, POP, PUSH, SETMOUSEMODE, SHIFT`
<plotapproximationmode>	`AUTOMATIC, NONCURRENTALWAYSAPPROX, ALLFRAMESALWAYSAPPROX`
<plottype>	`CARTESIAN3D, CARTESIAN2D, XYLINE, POLARLINE, SKETCH`
<pointerstyle>	`ALLDIRECTIONS, BOTTOM, LEFT, LEFTRIGHT, LOWERLEFT, LOWERRIGHT, RIGHT, TOP, UPDOWN, UPPERLEFT, UPPERRIGHT`

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<pointselection>	ALLPOINTS, NEARESTNPOINTS, OCTANTNPOINTS
<pointstoplot>	SURFACESONLY, ALL
<printerdriver>	HPGL, HPGL2, PS, EPS
<printrendertype>	VECTOR, IMAGE
<quickcolormode>	LINECOLOR, FILLCOLOR, TEXTCOLOR
<readdataoption>	NEW, APPEND, REPLACE
<relop>	<, >, <=, >=, ==, != (not equal to), <> (not equal to). GREATERTHAN, LESSTHAN, EQUALTO, NOTEQUALTO
<resizefilter>	TEXTUREFILTER, LANCZOS2FILTER, LANCZOS3FILTER, BOX-FILTER, TRIANGLEFILTER, BELLFILTER, BSPLINEFILTER, CUBICFILTER, MITCHELFILTER, GAUSSIANFILTER
<rgblegendorientation>	ORIENTRGB, ORIENTGBR, ORIENTBRG, ORIENTRBG, ORIENTBGR, ORIENTGRB
<rgbmode>	SPECIFYRGB, SPECIFYRG, SPECIFYRB, SPECIFYGB
<rotateaxis>	X, Y, Z, ALPHA, THETA, PSI, HORZROLLERBALL, VERTROLLERBALL, TWIST, ABOUTVECTOR
<rotateoriginlocation>	VIEWER, DEFINEDORIGIN
<rotationmode>	XYZAXIS, SPHERICAL, ROLLERBALL
<scope>	LOCAL, GLOBAL
<set>	[, *<setspecifier>, <setspecifier>*, ...,]
<setspecifier>	*<integer>, <integer>-<integer>[:<integer>]*
<sizeunits>	GRID, FRAME, POINT
<skipmode>	BYINDEX, BYFRAMEUNITS
<slicesource>	VOLUMEZONES, SURFACEZONES, SURFACESOFVOLUMEZONES, LINEARZONES
<sortby>	NONE, BYDEPENDENDTVAR, BYINDEPENDENTVAR, BYSPECIFIC-VAR
<standardcolormap>	SMRAINBOW, LGRAINBOW, MODERN, GRAYSCALE, TWOCOLOR
<stipplemode>	ALL, CRITICAL, NONE
<streamdirection>	FORWARD, REVERSE, BOTH
<streamtype>	SURFACELINE, VOLUMELINE, VOLUMERIBBON, VOLUMEROD, TWODLINE
<string>	"*<charactersequence>*", '*<charactersequence>*'[b]
<stylebase>	FACTORY, CONFIG
<subboundary>	ADD, ADDONLY, ALL, REMOVE

Table 7-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<sunrasterformat>	`OLDFORMAT, STANDARD, BYTEENCODED`
<surfacestoplot>	`BOUNDARYFACES, EXPOSEDCELLFACES, IPLANES, JPLANES, KPLANES, IJPLANES, JKPLANES, IKPLANES, IJKPLANES, ALL`
<textanchor>	`LEFT, CENTER, RIGHT, MIDLEFT, MIDCENTER, MIDRIGHT, HEADLEFT, HEADCENTER, HEADRIGHT`
<textboxtype>	`NONE, FILLED, HOLLOW`
<threedviewchange-drawlevel>	`FULL, TRACE`
<thetamode>	`DEGREES, RADIANS, ARBITRARY`
<tickdirection>	`IN, OUT, CENTERED`
<tiffbyteorder>	`INTEL, MOTOROLA`
<transformation>	`POLARTORECT, SPHERICALTORECT, RECTTOPOLAR, RECTTOSPHERICAL`
<translucency>	Valid integer from one to 99.
<twoddraworder>	`BYZONE, BYLAYER`
<valueblankcellmode>	`ALLCORNERS, ANYCORNER, PRIMARYCORNER`
<valueblankrelop>	`LESSTHANOREQUAL, GREATERTHANOREQUAL, NOTEQUALTO, GREATERTHAN, LESSTHAN, EQUALTO`
<valueformat>	`INTEGER, FLOAT, EXPONENT, BESTFLOAT, RANGEBESTFLOAT, SUPERSCRIPT, CUSTOMLABEL`
<valuelocation>	`AUTO, NODAL, CELLCENTERED`
<varloadmode>	`BYNAME, BYPOSITION`
<vectortype>	`TAILATPOINT, HEADATPOINT, MIDATPOINT, HEADONLY`
<viewmode>	`FIT, ZOOM, DATAFIT, AXISFIT, SETMAGNIFICATION, CENTER, TRANSLATE, LAST, COPY, PASTE, PUSH`
<workspaceviewmode>	`FITSELECTEDFRAMES, FITALLFRAMES, FITPAPER, MAXIMIZE, LASTVIEW, ZOOM, TRANSLATE`
<xyaxis>	`'X', 'Y'`

[a] Available in XY-plots only

[b] The only difference in using single quotes vs. double quotes for strings is that single quotes prevent the processing of the backslash character “\” (that is, **`\n`** inserts a newline, `\\` inserts the backslash itself).

7.2. Assignment Value Expressions

Simple values are literal constants such as 1, 3, 3.5, 2.5e17. Complex expressions are identified by an equation surrounded by `'('` and `')'` delimiters.

Expressions can be used within any layout or macro file and support all of the common operators and functions familiar to most C and FORTRAN programmers.

Arithmetic operators include the common multiply, divide, add, and subtract (`*`, `/`, `+` and `-`), as well as a few others (`^` and `**`) that are worth noting. The raise operator (`^`, or `**`) returns the result of raising the first number by the second.

Expressions may also contain macro variables and an assortment of useful functions and constants. Following are tables of supported functions and constants and a short explanation for each:

Table 7-2. Functions supported by Tecplot.

abs(*x*)	Absolute value of x.
acos(*x*)	Arc cosine of x between -1 and 1. Return an angle between 0 and p radians.
asin(*x*)	Arc sine of x between -1 and 1. Return an angle between -p/2 and p/2 radians.
atan(*x*)	Arc tangent of x. Return an angle between -p and p radians.
atan2(*y*,*x*)	Arc tangent of y/x . Return an angle between -p and p radians.
ceil(*x*)	Smallest integer larger than or equal to x.
cos(*x*)	Cosine of x in radians.
cosh(*x*)	Hyperbolic cosine of x.
exp(*x*)	Exponential of x.
floor(*x*)	Largest integer smaller than or equal to x.
frac(*x*)	Fractional part of x.
int(*x*)	Integer part of x.
log(*x*)	Natural logarithm of x.
log10(*x*)	Logarithm to the base 10 of x.
max(*x*,*y*)	Larger of x or y.
min(*x*,*y*)	Smaller of x or y.
pow(*x*,*y*)	xy.
sin(*x*)	Sine of x in radians.
sinh(*x*)	Hyperbolic sine of x.
sqrt(*x*)	Square root of x.
tan(*x*)	Tangent of x in radians.

Table 7-2. Functions supported by Tecplot.

tanh(*x*)	Hyperbolic tangent of x.

Constants are also supported, as listed in the following table.

Table 7-3. Constants supported by Tecplot.

BASEe	Natural logarithm base e.
DEG	Degrees per radian.
GAMMA	Euler-Mascheroni constant.
PHI	Golden ratio: $(\sqrt{5}+1)/2$.
PI	p.
RAD	Radians per degree.

The following table shows the operator precedence and associativity. Operators with higher precedence are listed in the higher rows of the table, while operators that are in the same row have the same precedence. The associativity describes how an operator associates with its operand.

Table 7-4. Operator precedence and associativity.

Operator Type	Operators	Associativity
Expression	*( )*	Left to right.
Power	^ **	Right to left.
Unary	- + !	Right to left.
Multiplicative	* /	Left to right.
Additive	+ -	Left to right.
Relational	> >= < <= == !=	Left to right.
Logical AND	&&	Left to right.
Logical OR	||	Left to right.
Conditional	? :	Right to left.

Unlike C, relational expressions do not evaluate to 0 or 1, instead, they evaluate to true or false. As such, they may only be used with other logical operators, or with the conditional operator.

Examples of common expressions used in the Tecplot macro language follow (note that all expressions evaluate to a simple, <*dexp*>, value):

```
$!If (|b|^2) > (4*|a|*|c|)
   $!If |a| > 0.0
      $!VarSet |root1| = (-|b| + sqrt(|b|^2 - 4*|a|*|c|) / (2*|a|))
      $!VarSet |root2| = (-|b| - sqrt(|b|^2 - 4*|a|*|c|) / (2*|a|))
   $!EndIf
```

```
$!EndIf

$!VarSet |area| = (PI*|r|**2)
```

In addition to the more common operators mentioned above, some relational and logical operators are provided to form compound expressions. A relation, <*relation*>, may be constructed and used in conjunction with the conditional operator (**?** and **:**) to form compound expressions. The conditional operator (**?** and **:**) has the following syntax:

<*relation*> **?** <*expression if true*> **:** <*expression if false*>

where:

- <*relation*> is a conditional statement that evaluates to true or false, and is formed by any two subexpressions which are compared to one another with one of the relational operators (`>`, `>=`, `<`, `<=`, `==`, `!=`) in combination with zero or more of the logical operators: `logical Not (!)`, logical *And* (`&&`), and logical *Or* (`||`).
- <*expression if true*> is the <*expression*> that is evaluated if the <*relation*> condition evaluates to **TRUE**.
- <*expression if false*> is the <*expression*> that is evaluated if the <*relation*> condition evaluates to **FALSE.**

Examples of compound expressions used in the Tecplot macro language follow (note that all compound expressions evaluate to a simple, <*dexp*>, value):

```
$!VarSet |value| = (|stress| > |cutoff| ? |cutoff| : |stress|)
$!VarSet |value| = (|x| < 1.5 && |y| <= 5.5 ? |x|^6 : (|x|+|y|)^3.2)
$!VarSet |root| = (|b|^2 > 4*|a|*|c| && |a| > 0.0 ? -|b| + sqrt(|b|^2 -
                  4*|a|*|c|) / (2*|a|) : 0)
```

It is important not to confuse an expression's relation, <*relation*>, that controls the evaluation of a compound expression, with the conditional expression, <*conditionalexp*>, that controls the execution of control commands such as **$!IF** and **$!WHILE**.

For example, the following is a valid macro command since it has a valid expression syntax and a valid control command syntax:

```
$!If |a| > (PI*|r|^2)
    ...
$!EndIf
```

The following is also a valid macro command because, like the last example, it has a valid expression syntax and a valid control command syntax:

```
$!If (|a|^2) == (|b| > 5 ? 1 : 0)
```

```
     ...
$!EndIf
```

The following is not a valid macro command since it has an invalid expression syntax and consequently an invalid control command syntax:

```
$!If (|a| > PI*|r|^2)
    ...
$!EndIf
```

As with the invalid example above, if Tecplot encounters a relation, <*relation*>, within an expression, <*expression*> (enclosed within `(` and `)` delimiters), it expects to find the conditional operator (`?` and `:`) and the two required expressions following the specified relation.

CHAPTER 8 Macro Variables

Macro variables are identified by a sequence of characters surrounded by vertical bars (“ | ”). Some examples are:

```
|myvariable|
                |loop|
                |1|
|$HOME|
```

Macro variables can be placed anywhere within a macro command. Upper case and lower case characters are treated the same. For example `|ABC|` and `|aBc|` represent the same variable.

Macro variables will be expanded to their value at the time the macro statement is processed.

Example: The following macro commands will result in a rotation of the data about the X-axis by 10 degrees:

```
$!VARSET |a1| = 10
$!ROTATE X
  ANGLE = |a1|
```

8.1. Internal Variables

The following table lists variables that are maintained by Tecplot which may be referenced by macro commands.

Variables	Notes
`|AUXDATASET|`	Retrieve auxiliary data from a data set. |AUXDATASET:Reynolds| would retrieve auxiliary data “Reynolds”
`|AUXFRAME|`	Retrieve auxiliary data from a frame. |AUXFRAME:Byron| would retrieve auxiliary data “Byron” from the current frame.
`|AUXZONE|`	Retrieve auxiliary data from a zone. |AUXZONE[3]:BC| would retrieve auxiliary data "BC" from zone 3 only.
`|AXISMAXA|`	Maximum value of current Theta-axis range.
`|AXISMAXR|`	Maximum value of current R-axis range.
`|AXISMAXX|`	Maximum value of current X-axis range.

Variables	Notes
|AXISMAXY|	Maximum value of current Y-axis range.
|AXISMAXZ|	Maximum value of current Z-axis range.
|AXISMINA|	Minimum value of current Theta-axis range.
|AXISMINR|	Minimum value of current R-axis range.
|AXISMINX|	Minimum value of current X-axis range.
|AXISMINY|	Minimum value of current Y-axis range.
|AXISMINZ|	Minimum value of current Z-axis range.
|BYTEORDERING|	Returns INTEL or MOTOROLA
|COLORMAPDYNAMIC|	Returns one if the color map is dynamic, zero if static.
|DATASETFNAME|	Returns data set file name.
|DATASETTITLE|	The title of the data set, or “No Data Set” if a dataset does not exist.
|DATE|	Returns the date in the form of 31 Jan 1998.
|ENDSLICEPOS|	Position of end slice.
|EXPORTISRECORDING|	Returns YES/NO to help macros complete record commands in proper order.
|FRAMENAME|	Returns the name of the current frame
|INBATCHMODE|	Returns one if Tecplot is in batch mode, zero if in interactive mode.
|ISDATASETAVAILABLE|	Returns 1 if a data set exists, and 0 if otherwise
|ISOSURFACELEVEL|	Returns the current iso-surface’s iso-value. The intrinsic must use array notation, meaning that |ISOSURFACE[2]| returns the value for the second iso-surface.
|LAYOUTFNAME|	Returns the current layout file name.
|LOOP|	Innermost loop counter.
|MACROFILEPATH|	Path to the directory containing the most recently opened macro file.
|MAXA|	Maximum value for Angle variable for polar line plots, calculated from the lowest numbered active polar line mapping.
|MAXB|	Maximum value for blanking variable. If the plot is 2D or 3D Cartesian, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MAXC|	Maximum value for contour variable. If the plot is 2D or 3D Cartesian, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.

Variables	Notes
|MAXI|	I-dimension for the lowest numbered active zone for 2D or 3D Cartesian plots. For line plots this represents the maximum I-value for the zone assigned to the lowest numbered active line mapping. For finite-element data, this represents the number of the nodes in the lowest order zones.
|MAXJ|	J-dimension for the lowest numbered active zone for 2D and 3D Cartesian plots. For line plots this represents the maximum J-value for the zone assigned to the lowest numbered active line mapping. For finite-element data, the number of elements in the lowest numbered active zone.
|MAXK|	K-dimension for the lowest numbered active zone for 2D and 3D Cartesian plots. For line plots this represents the maximum K-value for the zone assigned to the lowest numbered active line mapping. For finite-element data, this shows the number of nodes per element for the lowest numbered active zone.
|MAXR|	Maximum value of the R variable for polar line plots, calculated from the lowest numbered active polar line plot.
|MAXS|	Maximum value for scatter sizing variable for the currently active zones.
|MAXU|	Maximum value for variable assigned to the X-vector component for the currently active zones.
|MAXV|	Maximum value for variable assigned to the Y-vector component for the currently active zones.
|MAXV*nn***|**	Maximum value of variable *nn*.
|MAXVAR|	Returns the maximum values of the specified variable. It is indexed by array notation, meaning that a call of |MAXVAR[2]| gives the maximum value of the second variable.
|MAXW|	Maximum value for variable assigned to the Z-vector component for the currently active zones.
|MAXX|	Maximum value for variable assigned to the X-axis. If the plot is 2D or 3D Cartesian, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MAXY|	Maximum value for variable assigned to the Y-axis. For 2D or 3D Cartesian plots, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MAXZ|	Maximum value for variable assigned to the Z-axis for the currently active zones.
|MINA|	The minimum value for the Angle variable for polar line plots, calculate from the lowest numbered active polar line mapping.

Variables	Notes
|MINB|	Minimum value for blanking variable. For 2D or 3D Cartesian plots, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MINC|	Minimum value for contour variable. For 2D or 3D Cartesian plots, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MINS|	Minimum value for scatter sizing variable for the currently active zones.
|MINU|	Minimum value for variable assigned to the X-vector component for the currently active zones.
|MINV|	Minimum value for variable assigned to the Y-vector component for the currently active zones.
|MINV*nn***|**	Minimum value of variable *nn*.
|MINVAR|	Returns the minimum values of the specified variable. It is indexed by array notation, meaning that a call of |MINVAR[4]| gives the minimum value of the fourth variable.
|MINW|	Minimum value for variable assigned to the Z-vector component for the currently active zones.
|MINX|	Minimum value for variable assigned to the X-axis. For 2D or 3D Cartesian plots, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MINY|	Minimum value for variable assigned to the Y-axis. For 2D or 3D Cartesian plots, the value is calculated from the current set of active zones. For line plots, the value is calculated from the zone assigned to the lowest numbered active line mapping.
|MINZ|	Minimum value for variable assigned to the Z-axis for the currently active zones.
|NUMFRAMES|	Number of frames.
|NUMLINEMAPS|	Number of line maps assigned to the current frame.
|NUMPLANES|	Returns number of graphics bit-planes
|NUMVARS|	Number of variables in current data set.
|NUMZONES|	Number of zones in current data set.
|OPSYS|	Returns 1=UNIX, 2=DOS.
|PAPERHEIGHT|	Returns height of paper, that is, the white area of the Tecplot work area.
|PAPERSIZE|	Returns size of paper.
|PAPERWIDTH|	Returns the width of the paper.

Variables	Notes
|PLATFORM|	Returns name of platform, such as SGI or Windows.
|PLOTTYPE|	Zero = Sketch, one = XY, two = 2D, three = 3D, four = Polar line plots.
|PRINTFNAME|	Returns the file name of the last file sent for printing.
|SLICEPLANETYPE|	Plane type to which slices are assigned.
|STARTSLICEPOS|	Position of first slice.
|STREAMSTARTPOS|	Streamtrace starting position in X, Y, Z coordinates, given in the form of 0.5, 3.2 5.6.
|STREAMTYPE|	The streamtrace type such as "Surface Line", or "Surface Ribbon"
|TECHOME|	Path to the Tecplot home directory.
|TECPLOTVERSION|	Currently returns 100.
|TIME|	Returns the current time in the form of 12:15:28
|VARNAME|	Returns the name of a specified variable. This command uses array notation, so |VARNAME[3]| will return the name of the third variable.
|ZONEMESHCOLOR|	Returns the color of a particular zone mesh. Uses array notation.
|ZONENAME|	Returns the name of a specific zone. Uses array notation.

8.2. System Environment Variables

System environment variables can be accessed directly from within Tecplot by preceding an environment variable name with a "**$**" and surrounding it with vertical bars ("**|**"). Using environment variables within Tecplot adds another degree of flexibility to macros by taking advantage of each user's customized environment.

If an environment variable is missing, an error is generated and macro processing is terminated.

8.2.1. Example 1

To compare a macro variable with an environment variable:

```
$!IF |SESSION_COEFF| == |$DEFAULT_COEFF|
    # (perform some default processing here)
$!ENDIF
```

Where the **DEFAULT_COEFF** environment variable was set to some specified value of type double before starting Tecplot.

8.2.2. Example 2

To create a string from an environment variable:

```
$!VARSET |AUTHOR| = "Author: |$LOGNAME|"
```

8.3. User Defined Variables

User-defined variables are written using the macro variable name surrounded by vertical bars ("|"). The variable name can be up to 32 characters in length. If a macro variable is defined (using the **$!VARSET** command) and it is named the same as an existing internal macro variable, then the user-defined variable takes precedence and the internal value is not effected. The internal macro variable can be recovered if you remove the user-defined variable using **$!REMOVEVAR**.

8.4. Assigning Values to Macro Variables

The **$!VARSET** command is used to assign a value to a macro variable. The **$!VARSET** command has the following syntax:

```
$!VARSET <macrovar> <op> <double>
```

where *<op>* can be one of **=**, **-=**, **+=**, ***=**, or **/=**.

Examples:

Example 1: Add 2 to the macro variable **|ABC|:**

```
$!VARSET |ABC| += 2
```

Example 2: Set **|ABC|** to be equal to 37:

```
$!VARSET |ABC| = 37
```

Example 3: Multiply **|ABC|** by 1.5:

```
$!VARSET |ABC| *= 1.5
```

8.5. Assigning a String to a Macro Variable

Macro variables can be assigned to strings as well as to values. When using strings, only the "**=**" operator may be used.

Example: Assign the string **"myfile.plt"** to the variable **|FNAME|**. Use **|FNAME|** in the **$!READDATASET** command:

$!VARSET |FNAME| = "myfile.plt"

$!READDATASET "|FNAME|"

Note that double quotes (**"**) had to be used in the **$!READDATASET** command even though **|FNAME|** represents a string.

8.6. Replacement Text Use

You can assign replacement text to a macro variable. This is useful for handling cases where a macro variable may be not be initialized. A macro variable with |**AAAA:=XXXXX**| will produce **XXXXX** if **AAAA** is not defined. This does not work with intrinsic variables.

Example: Read in a data file assigned to the variable **FNAME**. If **FNAME** is unassigned, read in **"t.dat"**:

$!READDATASET "|FNAME:=t.dat|"

"|FNAME:=t.dat|"

8.7. Macro Function Variables

Macro function variables are written using a number *n*, surrounded by vertical bars (“ | ”). The number represents the *n*th parameter from the **$!RUNMACROFUNCTION** command.

Examples:

Example 1: The following commands define a macro function that uses two parameters and a command to run the macro function. The first parameter to the macro function is the amount to rotate about the X-axis and the second parameter is the amount to rotate about the Y-axis:

The command to run the macro function will cause a rotation of 10 degrees about the X-axis and 20 degrees about the Y-axis.

```
#!MC 1000
$!MACROFUNCTIONNAME = "3D Rotation Animation"
$!EXPORTSETUP EXPORTFORMAT = AVI
$!EXPORTSETUP IMAGEWIDTH = 546
$!EXPORTSETUP EXPORTFNAME = "|1|AxisRotation.avi"
$!EXPORTSTART
$!LOOP |2|
  ANGLE = 3
  ROTATEORIGINLOCATION = DEFINEORIGIN
$!REDRAW
$!EXPORTNEXTFRAME
$!ENDLOOP
```

```
$!EXPORTFINISH
$!ENDMACROFUNCTION
$!RUNMACTOFUNCTION "3D Rotation Animation" {"Theta", 6, 30}
```

Example 2: The following commands define a macro function that opens two layout files:

```
$!MACROFUNCTION
                  NAME = "OL2"
$!OPENLAYOUT "|1|"
$!OPENLAYOUT "|2|"
                  APPEND = TRUE
$!ENDMACROFUNCTION
                  .
                  .
                  .
$!RUNMACROFUNCTION "OL2" ("g1.lay","g2.lay")
```

8.8. Using Formats in Macro Variables

When a macro variable is expanded and the macro variable is a numeric value, it is expanded using a “best float” format. It tries to make the number look as simple as possible while still retaining as much accuracy as possible. If you want the number to be formatted in a specific way then you can include C-style number formatting strings in the macro variable specification. The syntax for including a format string is:

```
|macrovariable%formatstring|
```

Example 1: Suppose you want to pause a macro and display the message **"Maximum contour value is:** *xxxxxx*" where *xxxxxx* only has two digits to the right of the decimal place. You would use:

```
$!Pause "Maximum contour value is: |MAXC%.2f|"
```

If **|MAXC|** currently has a value of 356.84206 then the dialog would show:

```
"Maximum contour value is: 356.84"
```

Example 2: If, in the above example, you wanted to use exponential format you could use:

```
$!Pause "Maximum contour value is: |MAXC%12.6e|"
```

Here the result would be:

```
"Maximum contour value is: 3.568421e+02"
```

CHAPTER 9 Raw Data

Some macro commands contain a “raw data” section. A raw data section is defined by using the keyword **RAWDATA** followed by the raw data values unique to the macro command. Most raw data sections start with a single count value which represents the number of blocks of raw data followed by the blocks of raw data themselves. The following table lists the raw data sections found in Tecplot macros.

Raw Data Name	Value Type(s) per Block	Notes
<addoncommandrawdata>	*<string>*	Each line of the **RAWDATA** section contains an arbitrary text string. The only requirement is that the character sequence “$!” (a dollar sign followed by an exclamation mark) cannot appear anywhere in the section. Comments can be inserted by using # (the octothorp). If encountered, everything to the right of the # (including the # itself) will be ignored.
<colormaprawdata>	*<integer>* *<integer>* *<integer>*	Red. Green. Blue.
<contourlevelrawdata>	*<dexp>*	Contour level.
<geometryrawdata> *(Line segment geometry)*	*<xyrawdata>*	Each block contains a block of *<xyrawdata>,* which forms a single polyline within the geometry.
<geometryrawdata> *(3D Line segment)*	*<xyzrawdata>*	Each block contains a block of *<xyzrawdata>,* which forms a single polyline within the geometry.
<geometryrawdata> (circle)	*<dexp>*[a]	Only one value supplied. Value is the radius.
<geometryrawdata> (ellipse)	*<dexp>*[a] *<dexp>*[a]	Two values supplied. Values are RX and RY.
<geometryrawdata> (rectangle)	*<dexp>*[a] *<dexp>*[a]	Two values supplied. Values are width and height.
<geometryrawdata> (square)	*<dexp>*[a]	Only one value supplied. Value is the width.
<xyrawdata>	*<dexp>* *<dexp>*	X. Y.
<xyzrawdata>	*<dexp>* *<dexp>* *<dexp>*	X. Y. Z.

[a] A count value does not precede the raw data in this case.

Examples:

Example 1: Raw data for a circle with radius equal to 1.7:

```
RAWDATA
1.7
```

Example 2: Raw data for a line segment geometry with two segments. Segment 1 has 4 points and segment 2 has 3 points:

```
RAWDATA
2
4
1.5 2.2
1.7 2.4
1.9 2.8
2.1 3.0
3
1.1 1.7
1.2 1.9
1.3 2.0
```

Example 3: Raw data to define five contour levels:

```
RAWDATA
5
1.5
2.6
3.7
4.9
5.5
```

Example 4: Raw data to define three RGB values:

```
RAWDATA
3
0   0   0
45  100 100
90  200 200
```

Example 5: For greater control of contour levels in a macro, set the levels with RAWDATA. This example allows you to choose the number of levels, then sets new levels based on the minimum and maximum values of the current contour variable.

```
$!FIELDLAYERS SHOWCONTOUR = YES
$!Drawgraphics No
$!GLOBALCONTOUR 1  VAR = 4
$!PromptforTextString |numlevels|
 Instructions = "Enter the number of contour levels."
$!Varset |Delta| = ((|maxc| - |minc|)/|numlevels|)

$!CONTOURLEVELS DELETERANGE
  CONTOURGROUP = 1
  RANGEMIN = |minc|
  RANGEMAX = |maxc|
$!Varset |newlevel| = (|minc| + |delta|/2)

$!Loop |numlevels|
$!CONTOURLEVELS ADD
  CONTOURGROUP = 1
  RAWDATA
  1
  |newlevel|

$!Varset |newlevel| += |Delta|
$!Endloop
$!Drawgraphics Yes
$!REDRAW
```

CHAPTER 10 Macro Language Limitations

The only macro control commands allowed in stylesheets and layout files are:

$!VARSET and **$!REMOVEVAR**

The only SetValue command allowed in color map files is:

$!COLORMAP

Layout files, stylesheet files and colormap files cannot contain any of the following commands:

$!OPENLAYOUT
$!READSTYLESHEET
$!LOADCOLORMAP

Only SetValue macro commands are allowed in the Tecplot configuration file.

The $!LIMITS command can be used only in the Tecplot configuration file.

The $!FIELD and **$!LINEMAP** commands may be used in the configuration file but they may not specify an individual zone or line map. This special use of **$!FIELD** and **$!LINEMAP** allows you to change the default attributes for all zones and line mappings when they are initialized in Tecplot.

The file name referenced in the **$!INCLUDEMACRO** command cannot use Tecplot macro variables.

Size limitations:

Maximum number of nested macro function calls	10
Maximum number of nested macro loops	10
Maximum number of nested While-EndWhile loops	Unlimited.
Maximum number of nested If-EndIf loops	Unlimited.
Maximum number of nested macro includes	5
Maximum number of macro commands	200,000
Maximum number of parameters per macro function	20

Maximum number of characters in macro variable name	31
Maximum number of characters in macro function name	Unlimited.
Maximum number of macro variables	400

PART II *Binary Data*

CHAPTER 11 Writing Binary Data for Loading into Tecplot

This chapter is intended only for advanced users of Tecplot who have a solid background in UNIX or Windows and application programming. Support for topics discussed in this chapter may be limited. Regular technical support is not intended to help you program your application to use the direct data file capabilities of Tecplot.

Data files for Tecplot are commonly created as output from an application program. These files are most often in ASCII format, and are then converted to a binary format with Preplot.

Included with your distribution of Tecplot is a library that contains utility functions that you can link with your application program to create binary data files directly, bypassing the use of ASCII files. This allows for fewer files to manage, conserves on disk space, and saves the extra time required to convert the files.

In UNIX, the utility functions discussed below are available in the library archive **`tecio.a`** which is located in the **`lib`** sub-directory of the Tecplot Home Directory. Under Windows, this library is called **`TecIO.dll`** and is located in the **`bin`** sub-directory. Instructions on compiling and linking using the **`TECIO`** library can be found in the **`readme.doc`** file in the **`util/tecio`** sub-directory under the **`TECHOME`** directory.

Tecplot 10 introduces a new set of TECIO functions to take full advantage of the new capabilities it offers. Each of these functions has a suffix of "100" to differentiate it from previous editions. Please note that all existing, Version 9, TECIO functions still exist and are supported for backward compatibility.

11.1. Function Summary

The following functions are available from the **TECIO** archive. For historical reasons, these functions have a FORTRAN flavor to them, both in how they are named and the way in which the parameters are passed.

Tecplot Version 10 TECIO Functions:

- **TECINI100**: Initialize the process of writing a binary data file.
- **TECZNE100**: Write information about the next zone to be added to the data file.
- **TECDAT100**: Write an array of data to the data file.
- **TECNOD100**: Write an array of node data to the data file.
- **TECLAB100**: Write a custom label record to the data file.
- **TECGEO100**: Write a geometry record to the data file.
- **TECTXT100**: Write a text record to the data file.
- **TECFIL100**: Switch output context to a different file.
- **TECEND100**: Close the data file.
- **TECUSR100:** Write a character string to the data file in a USERREC record.
- **TECAUXSTR100**: Write auxiliary data for the data set to the data file.
- **TECZAUXSTR100**:Write auxiliary data for the current zone to the data file.
- **TECFACE100:** Write the face connections for the current zone to the data file.

Existing Tecplot TECIO Functions:

- **TECINI:** Initialize the process of writing a binary data file.
- **TECZNE**: Write information about the next zone to be added to the data file.
- **TECDAT:** Write an array of data to the data file.
- **TECNOD:** Write an array of node data to the data file.
- **TECLAB**: Write a custom label record to the data file.
- **TECGEO**: Write a geometry record to the data file.
- **TECTXT**: Write a text record to the data file.
- **TECFIL**: Switch output context to a different file.
- **TECEND**: Close the data file.

11.2. Binary Data File Function Calling Sequence

Multiple data files can be written to at the same time. For a given file, the binary data file functions must be called in a specific order.

The correct order is as follows:

```
TECINI100
  TECAUXSTR100
  TECZNE100      (One or more to create multiple zones)
    TECDAT100    (One or more to fill each zone)
    TECNOD100    (One for each finite element zone)
    TECFACE100   (One for each zone with face connections)
    TECZAUXSTR100
  TECLAB100
  TECGEO100
  TECTXT100
  TECUSR100
TECEND
```

Section 11.3, “Writing to Multiple Binary Data Files,” explains how you can use the **TECFIL100** function along with the above functions to write to multiple files at the same time.

The **TECZNE100, TECLAB100**, **TECGEO100, TECAUXSTR100** and **TECTXT100** functions can be called anywhere between the **TECINI100** and **TECEND100** functions. **TECDAT100** and **TECNOD100** (for finite-element data only) must be called immediately after the **TECZNE100** function call. **TECFACE100** (where face connections were indicated in the call to **TECZNE100)** must be called immediately after **TECNOD100** (for finite-element data) or **TECZNE100** (for ordered data). **TECZAUXSTR100** must be called following the **TECZNE100** call for the zone with which the auxiliary data is associated.

11.3. Writing to Multiple Binary Data Files

Each time **TECINI100** is called it sets up a new file “context.” For each file context you must maintain the order of the calls as described in the previous section. The **TECFIL100** function is used to switch between file contexts. Up to 10 files can be written to at a time. **TECFIL100** can be called almost anywhere after **TECINI100** has been called. The only parameter to **TECFIL100**, an integer, *n*, shifts the file context to the *n*th open file where the files are numbered relative to the order of the calls to **TECINI100**. See Section 11.7.3, “Complex Example (FORTRAN),” and 11.7.4, “Complex Example (C),” at the end of this chapter for an example of how to use the **TECFIL100** function to write to multiple files.

11.4. Character Strings in FORTRAN

All character string parameters in FORTRAN must terminate with a null character. This is done by concatenating **char(0)** to the end of a character string.

For example, to send the character string “Hi Mom” to a function called **A**, the syntax would be:

```
I=A("Hi Mom"//char(0))
```

11.5. Boolean Flags

Integer parameters identified as "flags" indicate boolean values. Pass 1 for true, and 0 for false.

11.6. Binary Data File Function Reference

This section describes each of the **TECIO** functions in detail.

TECAUXSTR100

Summary: Writes auxiliary data for the data set to the data file. The function may be called any time between **TECINI100** and **TECEND100**. Auxiliary data may be used by text, macros, equations (if it is numeric) and add-ons. It may be viewed directly in the AuxData page of the Data Set Information dialog.

FORTRAN Syntax:

```
      INTEGER*4 FUNCTION TECAUXSTR100(Name,
   &                    Value)
      CHARACTER*(*) Name
      CHARACTER*(*) Value
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECAUXSTR100(char *Name,
                      char *Value)
```

Return Value: **0** if successful, **-1** if unsuccessful.

Parameters: *Name*

The name of the auxiliary data. If this duplicates an existing name, the value will overwrite the existing value. Must be a null-terminated character string.

Value

The value to assign to the named auxiliary data. Must be a null-terminated character string.

TECDAT100

Summary: Writes an array of data to the data file.

The following table describes the order the data must be supplied given different zone types (IsBlock is a parameter supplied to TECZONE100):

Zone Type	Variable Location	IsBlock	Number of Values Supplied	Order
Ordered	Nodal	1	IMax* JMax* KMax* NumVars	I varies fastest, then J, then K, then V
Ordered	Nodal	0	IMax* JMax* KMax* NumVars	V varies fastest, then I, then J, then K
Ordered	Cell Centered	1	(IMax-1)* (JMax-1)* (KMax-1)* NumVars	I varies fastest, then J, then K, then V
Ordered	Cell Centered	0	Not allowed	
Finite Element	Nodal	1	IMax (i.e. NumPts) * NumVars	N varies fastest, then V
Finite Element	Nodal	0	IMax (i.e. NumPts) * NumVars	V varies fastest, then N
Finite Element	Cell Centered	1	JMax (i.e. NumElements) * NumVars	E varies fastest, then V
Finite Element	Cell Centered	0	Not allowed	

Note that if any variables are cell centered then the data must be supplied in block format thus the IsBlock parameter in TECZONE100 MUST be set to 1

TECDAT100 allows you to write your data in a piecemeal fashion in case it is not contained in one contiguous block in your program. Enough calls to **TECDAT100** must be made that the correct number of values are written for each zone and that the aggregate order for the data is correct.

In the above summary, *NumVars* is based on the number of variable names supplied in a previous call to **TECINI100**.

FORTRAN Syntax:

```
  INTEGER*4 FUNCTION TECDAT100(N,
&                             Data,
&                             IsDouble)
  INTEGER*4 N
  REAL or DOUBLE PRECISION Data(1)
  INTEGER*4 IsDouble
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECDAT100(INTEGER4 *N,
                   void *Data,
                   INTEGER4 *IsDouble);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *N*

Pointer to an integer value specifying number of values to write.

Data

Array of single or double precision data values.

IsDouble

Pointer to the integer flag stating whether the array *Data* is single (`0`) or double (`1`) precision.

TECEND100

Summary: *Must* be called to close out the current data file. There must be a corresponding **TECEND100** for each **TECINI100**.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECEND100()
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECEND100();
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: None.

TECFACE100

Summary: Writes face connections for the current zone to the file. This function must be called after **`TECNOD100`**, and may only be called if a non-zero value of *NumFaceConnections* was used in the previous call to **`TECZNE100`**.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECFACE100(FaceConnections)

INTEGER*4 FACECONNECTIONS
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECFACE100(INTEGER4 *FaceConnections);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *FaceConnections*

The array that specifies the face connections. The array must be dimensioned **`(L, NumFaceConnections)`**, where L is determined by the type of face connection specified by the **`FaceNeighborMode`** parameter to **`TECZNE100`**:

FaceNeighbor Mode	# Values	Data
LocalOneToOne	3	cz,fz,cz
LocalOneToMany	nz+4	cz,fz,oz,nz,cz1,cz2,...,czn
GlobalOneToOne	4	cz,fz,ZZ,CZ
GlobalOneToMany	2*nz+4	cz,fz,oz,nz,ZZ1,CZ1,ZZ2,CZ2,...,ZZn,CZn

Where:

cz = cell in current zone

fz = face of cell in current zone

oz = face obscuration flag (only applies to one-to-many):

0 = face partially obscured

1 = face entirely obscured

nz = number of cell or zone/cell associations (only applies to one-to-many)

ZZ = remote Zone

CZ = cell in remote zone

cz,fz combinations must be unique. Additionally, Tecplot assumes that with the one-to-one face neighbor modes a supplied cell face is entirely obscured by its neighbor. With one-to-many, the obscuration flag must be supplied. Faces that are not supplied with neighbors are run through Tecplot's auto face neighbor generator (FE only).

TECFIL100

Summary: Switch output context to a different file. Each time **TECINI100** is called, a new file "context" is switched to. This allows you to write multiple data files at the same time.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECFIL100(F)

INTEGER*4 F
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECFIL100(INTEGER4 *F);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *F*

Pointer to integer specifying file number to switch to. A value of 1 indicates a switch to the file opened by the first call to **TECINI100**.

TECGEO100

Summary: Writes a geometry to the data file.

FORTRAN Syntax:

```
 INTEGER*4 FUNCTION TECGEO100(XPos,
&                            YPos,
&                            ZPos,
&                            PosCoordMode,
&                            AttachToZone,
&                            Zone,
&                            Color,
&                            FillColor,
&                            IsFilled,
&                            GeomType,
&                            LinePattern,
&                            PatternLength,
&                            LineThickness,
&                            NumEllipsePts,
&                            ArrowheadStyle,
&                            ArrowheadAttachment,
&                            ArrowheadSize,
&                            ArrowheadAngle,
&                            Scope,
&                            Clipping,
&                            NumSegments,
&                            NumSegPts,
&                            XGeomData,
&                            YGeomData,
&                            ZGeomData,
&                            MFC)
 DOUBLE PRECISION XPos
 DOUBLE PRECISION YPos
 DOUBLE PRECISION ZPos
 INTEGER*4 PosCoordMode
 INTEGER*4 AttachToZone
 INTEGER*4 Zone
 INTEGER*4 Color
 INTEGER*4 FillColor
 INTEGER*4 IsFilled
 INTEGER*4 GeomType
 INTEGER*4 LinePattern
 DOUBLE PRECISION PatternLength
 DOUBLE PRECISION LineThickness
 INTEGER*4 NumEllipsePts
 INTEGER*4 ArrowheadStyle
 INTEGER*4 ArrowheadAttachment
 DOUBLE PRECISION ArrowheadSize
 DOUBLE PRECISION ArrowheadAngle
INTEGER*4 Scope

INTEGER*4 Clipping

 INTEGER*4 NumSegments
 INTEGER*4 NumSegPts
 REAL*4 XGeomData
 REAL*4 YGeomData
 REAL*4 ZGeomData
CHARACTER*(*) MFC
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECGEO(double *XPos,
                double *YPos,
                double *ZPos,
                INTEGER4 *PosCoordMode,
                INTEGER4 *AttachToZone,
                INTEGER4 *Zone,
                INTEGER4 *Color,
                INTEGER4 *FillColor,
                INTEGER4 *IsFilled,
                INTEGER4 *GeomType,
                INTEGER4 *LinePattern,
                double *PatternLength,
                double *LineThickness,
                INTEGER4 *NumEllipsePts,
                INTEGER4 *ArrowheadStyle,
                INTEGER4 *ArrowheadAttachment,
                double *ArrowheadSize,
                double *ArrowheadAngle,
                INTEGER4 *Scope,
                INTEGER4 *Clipping,
                INTEGER4 *NumSegments,
                INTEGER4 *NumSegPts,
                float *XGeomData,
                float *YGeomData,
                float *ZGeomData,
                char *MFC)
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *XPos*

Pointer to double value specifying the X-position or, for polar line plots, the Theta-position of the geometry.

YPos

Pointer to double value specifying the Y-position or, for polar line plots, the R-position of the geometry.

ZPos

Pointer to double value specifying the Z-position of the geometry.

PosCoordMode

Pointer to integer value specifying the position coordinate system.

0=Grid
1=Frame
4=Grid3D

AttachToZone

Pointer to integer flag to signal that the geometry is “attached” to a zone.

Zone

Pointer to integer value specifying the number of the zone to attach to.

Color

Pointer to integer value specifying the color to assign to the geometry.

0=Black
1=Red
2=Green
3=Blue
4=Cyan
5=Yellow
6=Purple
7=White

8=Custom1
9=Custom2
10=Custom3
11=Custom4
12=Custom5
13=Custom6
14=Custom7
15=Custom8

FillColor

Pointer to integer value specifying the color used to fill the geometry. See *Color* above.

IsFilled

Pointer to integer flag to specify if geometry is to be filled.

GeomType

Pointer to integer value specifying the geometry type.

0=2DLineSegments

1=Rectangle

2=Square

3=Circle

4=Ellipse

LinePattern

Pointer to integer value specifying the line pattern.

0=Solid

1=Dashed

2=DashDot

3=Dotted

4=LongDash

5=DashDotDot

PatternLength

Pointer to double value specifying the pattern length in frame units.

LineThickness

Pointer to double value specifying the line thickness in frame units.

NumEllipsePts

Pointer to integer value specifying the number of points to use for circles and ellipses. The value must be greater than 0.

ArrowheadStyle

Pointer to integer value specifying the arrowhead style.

0=Plain

1=Filled

2=Hollow

ArrowheadAttachment

Pointer to integer value specifying where to attach arrowheads.

0=None

1=Beginning

2=End

3=Both

ArrowheadSize

Pointer to double value specifying the arrowhead size in frame units.

ArrowheadAngle

Pointer to double value specifying the arrowhead angle in degrees.

Scope

Pointer to integer value specifying the scope. 0=global, 1=local.

Clipping

Specifies whether to clip the geometry (that is, only plot the geometry within) to the viewport or the frame. 0=ClipToViewport,1=ClipToFrame.

NumSegments

Pointer to integer value specifying the number of polyline segments.

NumSegPts

Array of integer values specifying the number of points in each of the *NumSegments* segments.

XGeomData

Array of floating-point values specifying the X-coordinates.

YGeomData

Array of floating-point values specifying the Y-coordinates.

ZGeomData

Array of floating-point values specifying the Z-coordinate.

MFC

Macro function command. Must be null terminated.

TECINI100

Summary: Initializes the process of writing a binary data file. This must be called *first* before any other **TECIO** calls are made. You may write to multiple files by calling **TECINI100** more than once. Each time **TECINI100** is called, a new file is opened. Use **TECFIL100** to switch between files.

FORTRAN Syntax:

```
  INTEGER*4 FUNCTION TECINI100(Title,
 &                                 Variables,
 &                                 FName,
 &                                 ScratchDir,
 &                                 Debug,
 &                                 VIsDouble)
```

```
CHARACTER*(*) Title
CHARACTER*(*) Variables
CHARACTER*(*) FName
CHARACTER*(*) ScratchDir
INTEGER*4 Debug
INTEGER*4 VIsDouble
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECINI100(char *Title,
          char *Variables,
          char *FName,
          char *ScratchDir,
          INTEGER4 *Debug
          INTEGER4 *VIsDouble);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *Title*

Title of the data set. *Must be null terminated.*

Variables

List of variable names. If a comma appears in the string it will be used as the separator between variable names, otherwise a space is used. *Must be null terminated.*

FName

Name of the file to create. Must be null terminated.

ScratchDir

Name of the directory to put the scratch file. Must be null terminated.

Debug

Pointer to the integer flag for debugging. Set to 0 for no debugging or 1 to debug.

VIsDouble

Pointer to the integer flag for specifying whether field data generated in future calls to **`TECDAT`** are to be written in single or double precision. Set to 0 for single precision or 1 for double.

TECLAB100

Summary: Write a set of custom labels to the data file.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECLAB100(Labels)
 CHARACTER*(*)  Labels
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECLAB100(char *Labels);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *Labels*

Character string of custom labels. Separate labels by a comma or space. For example, a set of custom labels for each day of the weeks is **`Sun Mon Tue Wed Thu Fri Sat`**.

TECNOD100

Summary: Writes an array of node data to the binary data file. This is the connectivity list for finite element zones.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECNOD100(NData)
INTEGER*4  NData(T, M)
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECNOD100(INTEGER4 *NData);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *NData*

Array of integers. This is the connectivity list, dimensioned **(***T***,** *M***)** (*T* moving fastest), where *M* is the number of elements in the zone and *T* is set according to the following list:

ELEMENT TYPE	*T*
Line Segment	2
Triangle	3

ELEMENT TYPE	*T*
Quadrilateral	4
Tetrahedral	4
Brick	8

TECTXT100

Summary: Writes a text record to the data file.

FORTRAN Syntax:

```
 INTEGER*4 FUNCTION TECTXT100(XOrThetaPos,
&                             YOrRPos,
&                             ZOrUnusedPos,
&                             PosCoordMode,
&                             AttachToZone,
&                             Zone,
&                             Font,
&                             FontHeightUnits,
&                             FontHeight,
&                             BoxType,
&                             BoxMargin,
&                             BoxLineThickness,
&                             BoxColor,
&                             BoxFillColor,
&                             Angle,
&                             Anchor,
&                             LineSpacing,
&                             TextColor,
&                             Scope,
&                             Clipping,
&                             Text,
&                             MFC)
 DOUBLE PRECISION XOrThetaPos
 DOUBLE PRECISION YOrRPos
 DOUBLE PRECISION ZOrUnusedPos,
 INTEGER*4 PosCoordMode
 INTEGER*4 AttachToZone
 INTEGER*4 Zone
 INTEGER*4 Font
 INTEGER*4 FontHeightUnits
 DOUBLE PRECISION FontHeight
 INTEGER*4 BoxType
```

```
DOUBLE PRECISION  BoxMargin
DOUBLE PRECISION BoxLineThickness
INTEGER*4 BoxColor
INTEGER*4 BoxFillColor
DOUBLE PRECISION Angle
INTEGER*4 Anchor
DOUBLE PRECISION LineSpacing
INTEGER*4 TextColor
INTEGER*4 Scope
INTEGER*4 Clipping
CHARACTER*(*) Text
CHARACTER*(*) MFC
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECTXT100(double *XOrThetaPos,
                   double *YOrRPosPos,
                   double *ZOrUnusedPos,
                   INTEGER4 *PosCoordMode,
                   INTEGER4 *AttachToZone,
                   INTEGER4 *Zone,
                   INTEGER4 *Font,
                   INTEGER4 *FontHeightUnits,
                   double *FontHeight,
                   INTEGER4 *BoxType,
                   double *BoxMargin,
                   double *BoxLineThickness,
                   INTEGER4 *BoxColor,
                   INTEGER4 *BoxFillColor,
                   double *Angle,
                   INTEGER4 *Anchor,
                   double *LineSpacing,
                   INTEGER4 *TextColor,
                   INTEGER4 *Scope,
                   INTEGER4 *Clipping,
                   char *Text,
                   char *MFC)
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *XOrThetaPos*

Pointer to double value specifying the X-position or Theta-position (polar plots only) of the text.

YOrRPos

Pointer to double value specifying the Y-position or R-position (polar plots only) of the text.

ZOrUnusedPos

Pointer to double value specifying the Z-position of the text.

PosCoordMode

Pointer to integer value specifying the position coordinate system.

0=Grid
1=Frame
4=Grid3D

AttachToZone

Pointer to integer flag for to signal that the text is “attached” to a zone.

Zone

Pointer to integer value specifying the zone number to attach to.

Font

Pointer to integer value specifying the font.

0=Helvetica
1=Helvetica Bold
2=Greek
3=Math
4=User-Defined
5=Times
6=Times Italic
7=Times Bold
8=Times Italic Bold
9=Courier
10=Courier Bold

FontHeightUnits

Pointer to integer value specifying the font height units.

0=Grid
1=Frame
2=Point

FontHeight

Pointer to double value specifying the font height.

BoxType

Pointer to integer value specifying the box type.

0=None

1=Filled

2=Hollow

BoxMargin

Pointer to double value specifying the box margin (in frame units).

BoxLineThickness

Pointer to double value specifying the box line thickness (in frame units).

BoxColor

Pointer to integer value specifying the color to assign to the box.

0=Black

1=Red

2=Green

3=Blue

4=Cyan

5=Yellow

6=Purple

7=White

8=Custom1

9=Custom2

10=Custom3

11=Custom4

12=Custom5

13=Custom6

14=Custom7

15=Custom8

BoxFillColor

Pointer to integer value specifying the fill color to assign to the box. (See *BoxColor*)

Angle

Pointer to double value specifying the text angle in degrees.

Anchor

Pointer to integer value specifying where to anchor the text.

0=Left

1=Center

2=Right

3=MidLeft

4=MidCenter

5=MidRight

6=HeadLeft

7=HeadCenter

8=HeadRight

LineSpacing

Pointer to double value specifying the text line spacing.

TextColor

Pointer to integer value specifying the color to assign to the text. (See *BoxColor*)

Scope

Pointer to integer value specifying the scope.

0=Global

1=Local

Clipping

Specifies whether to clip the geometry (that is, only plot the geometry within) to the viewport or the frame. 0=ClipToViewport,1=ClipToFrame.

Text

Character string representing text to display. Must be null terminated.

MFC

Macro function command. Must be null terminated.

TECUSR100

Summary: Writes a character string to the data file in a USERREC record. USERREC records are ignored by Tecplot, but may be used by add-ons.

FORTRAN Syntax:

```
    INTEGER*4 FUNCTION TECUSR100(S)
 CHARACTER*(*) S
```

C Syntax:

```
#include TECIO.h
```

```
INTEGER4 TECUSR100(CHAR *S);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *S*

The character string to write to the data file. Must be null-terminated.

TECZAUXSTR100

Summary: Writes an auxiliary data item for the current zone to the data file. Must be called after **`TECZNE100`** for the desired zone. Auxiliary data may be used by text, macros, equations (if it is numeric) and add-ons. It may be viewed directly in the AuxData page of the Data Set Information dialog.

FORTRAN Syntax:

```
    INTEGER*4 FUNCTION TECZAUXSTR100(Name, Value)
&
   CHARACTER*(*) Name
   CHARACTER*(*) Value
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECZAUXSTR100(char *Name,
                       char *Value);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *Name*

The name of the auxiliary data item. If a data item with this name already exists, its value will be overwritten. Must be a null-terminated character string.

Value

The auxiliary data value to be written to the data file. Must be a null-terminated character string.

TECZNE100

Summary: Writes header information about the next zone to be added to the data file. After **`TECZNE100`** is called, you must call **`TECDAT100`** one or more times (and then call **TECNOD100** if the data format is **FEBLOCK** or **FEPOINT**).

FORTRAN Syntax:

```
 INTEGER*4 FUNCTION TECZNE100(ZoneTitle,
&                             ZoneType,
&                             IMxOrNumPts,
&                             JMxOrNumElements,
&                             KMx,
&                              ICellMax,
&                              JCellMax,
&                              KCellMax,
&                              IsBlock,
&                              NumFaceConnections,
&                              FaceNeighborMode,
&                              ValueLocation,
&                              ShareVarFromZone
&                              ShareConnectivityFromZone)
 CHARACTER*(*) ZoneTitle
 INTEGER*4 ZoneType
 INTEGER*4 IMxOrNumPts
 INTEGER*4 JMxOrNumElements
 INTEGER*4 KMx
 INTEGER*4 ICellMax
 INTEGER*4 JCellMax
 INTEGER*4 KCellMax
 INTEGER*4 N
 INTEGER*4 M
 INTEGER*4 IsBlock
 INTEGER*4 NumFaceConnections
 INTEGER*4 FaceNeighborMode
 INTEGER*4 ValueLocation
 INTEGER*4 ShareVarFromZone
 INTEGER*4 ShareConnectivityFromZone
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECZNE100(char *ZoneTitle,
                   INTEGER4 *ZoneType,
                   INTEGER4 *IMxOrNumPts,
                   INTEGER4 *JMxOrNumElements,
                   INTEGER4 *KMx,
                   INTEGER4 *ICellMax,
                   INTEGER4 *JCellMax,
                   INTEGER4 *KCellMax,
                   INTEGER4 *IsBlock,
                   INTEGER4 *NumFaceConnections,
                   INTEGER4 *FaceNeighborMode,
                   INTEGER4 *ValueLocation,
                   INTEGER4 *ShareVarFromZone,
                   INTEGER4 *ShareConnectivityFromZone)
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *ZoneTitle*

The title of the zone. Must be null-terminated.

ZoneType:

The type of the zone: 0=ORDERED,1=FELINESEG,2=FETRIANGLE,3=FEQUADRILATERAL ,4=FETETRAHEDRON,5=FEBRICK

IMxOrNumPts:

For ordered zones, the number of nodes in the I index direction. For finite-element zones, the number of nodes.

JMxOrNumElements:

For ordered zones, the number of nodes in the J index direction. For finite-element zones, the number of elements.

KMx:

For ordered zones, the number of nodes in the K index direction. Not used for finite-element zones.

ICellMax:

For zones of type FEBRICK only, the number of cells logically connected in the I index direction.

JCellMax:

For zones of type FEBRICK only, the number of cells logically connected in the J index direction.

KCellMax:

For zones of type FEBRICK only, the number of cells logically connected in the K index direction.

IsBlock:

Indicates whether the data will be passed into TECDAT100 in BLOCK or POINT format. 0=POINT, 1=BLOCK.

NumFaceConnections:

The number of face connections that will be passed in routine TECFACE100.

FaceNeighborMode:

The type of face connections that will be passed in routine TECFACE100. 0=LocalOneToOne, 1=LocalOneToMany, 2=GlobalOneToOne, 3=GlobalOneToMany

ValueLocation:

The location of each variable in the data set. ValueLocation(I) indicates the location of variable I for this zone. 0=cell-centered, 1=node-centered. Pass null to indicate that all variables are node-centered.

ShareVarFromZone:

Indicates variable sharing. ShareVarFromZone(I) indicates the zone number with which variable I will be shared. This reduces the amount of data to be passed via TECDAT100. A value of 0 indicates that the variable is not shared. Pass null to indicate no variable sharing for this zone. You must pass null for the first zone in a data set (there is no data available to share).

ShareConnectivityFromZone:

For finite-element zones only, Indicates the zone number with which connectivity is shared. Pass 0 to indicate no connectivity sharing. You must pass 0 for the first zone in a data set.

The commands below are the old TECIO commands which still work for purposes of backwards compatibility. Note that in many cases, these functions take the same inputs as their Version 10 counterparts.

TECDAT

Summary: Writes an array of data to the data file.

If the *ZoneFormat* specified in **TECZNE** is **BLOCK**, the array must be dimensioned (*IMax*, *JMax*, *KMax*, *NumVars*) (FORTRAN syntax, where the first element moves the fastest).

If the *ZoneFormat* is **POINT**, the data must be dimensioned (*NumVars*, *IMax*, *JMax*, *KMax*).

If the *ZoneFormat* is **FEBLOCK**, then the data must be dimensioned (*NumPts*, *NumVars*).

If the *ZoneFormat* is **FEPOINT**, then the data must be dimensioned

(*NumVars*, *NumPts*).

TECDAT allows you to write your data in a piecemeal fashion in case it is not contained in one contiguous block in your program. Enough calls to **TECDAT** must be made that the correct number of values are written for each zone and that the aggregate order for the data is correct.

In the above summary, *NumVars* is based on the number of variable names supplied in a previous call to **TECINI**.

FORTRAN Syntax:

```
     INTEGER*4 FUNCTION TECDAT(N,
    &                         Data,
    &                         IsDouble)
     INTEGER*4 N
     REAL or DOUBLE PRECISION Data(1)
     INTEGER*4 IsDouble
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECDAT(INTEGER4 *N,
                void *Data,
                INTEGER4 *IsDouble);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *N*

Pointer to an integer value specifying number of values to write.

Data

Array of single or double precision data values.

IsDouble

Pointer to the integer flag stating whether the array *Data* is single (`0`) or double (`1`) precision.

TECEND

Summary: *Must* be called to close out the current data file. There must be a corresponding **TECEND** for each **TECINI**.

FORTRAN Syntax:

```
     INTEGER*4 FUNCTION TECEND()
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECEND();
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: None.

TECFIL

Summary: Switch output context to a different file. Each time **`TECINI`** is called, a new file "context" is switched to. This allows you to write multiple data files at the same time.

FORTRAN Syntax:

```
      INTEGER FUNCTION TECFIL(F)
      INTEGER*4 F
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECFIL(INTEGER4 *F);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *F*

Pointer to integer specifying file number to switch to.

TECGEO

Summary: Writes a geometry to the data file.

FORTRAN Syntax:

```
      INTEGER*4 FUNCTION TECGEO(XPos,
    &                           YPos,
    &                           ZPos,
    &                           PosCoordMode,
    &                           AttachToZone,
    &                           Zone,
    &                           Color,
    &                           FillColor,
    &                           IsFilled,
    &                           GeomType,
    &                                LinePattern,
    &                                PatternLength,
    &                                LineThickness,
    &                                NumEllipsePts,
    &                                ArrowheadStyle,
    &                                ArrowheadAttachment,
    &                                ArrowheadSize,
    &                                ArrowheadAngle,
    &                                Scope,
    &                                NumSegments,
    &                                NumSegPts,
    &                                XGeomData,
    &                                YGeomData,
    &                                ZGeomData,
    &                                MFC)
      DOUBLE PRECISION XPos
      DOUBLE PRECISION YPos
      DOUBLE PRECISION ZPos
      INTEGER*4 PosCoordMode
      INTEGER*4 AttachToZone
      INTEGER*4 Zone
      INTEGER*4 Color
      INTEGER*4 FillColor
      INTEGER*4 IsFilled
      INTEGER*4 GeomType
      INTEGER*4 LinePattern
      DOUBLE PRECISION PatternLength
      DOUBLE PRECISION LineThickness
      INTEGER*4 NumEllipsePts
      INTEGER*4 ArrowheadStyle
      INTEGER*4 ArrowheadAttachment
      DOUBLE PRECISION ArrowheadSize
      DOUBLE PRECISION ArrowheadAngle
      INTEGER*4 Scope
      INTEGER*4 NumSegments
      INTEGER*4 NumSegPts
      REAL*4 XGeomData
      REAL*4 YGeomData
      REAL*4 ZGeomData
    CHARACTER*(*) MFC
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECGEO(double *XPos,
                    double *YPos,
                    double *ZPos,
                    INTEGER4 *PosCoordMode,
                    INTEGER4 *AttachToZone,
                    INTEGER4 *Zone,
                    INTEGER4 *Color,
                    INTEGER4 *FillColor,
                    INTEGER4 *IsFilled,
                    INTEGER4 *GeomType,
                    INTEGER4 *LinePattern,
                    double *PatternLength,
                    double *LineThickness,
                    INTEGER4 *NumEllipsePts,
                    INTEGER4 *ArrowheadStyle,
                    INTEGER4 *ArrowheadAttachment,
                    double *ArrowheadSize,
                    double *ArrowheadAngle,
                    INTEGER4 *Scope,
                    INTEGER4 *NumSegments,
                    INTEGER4 *NumSegPts,
                    float *XGeomData,
                    float *YGeomData,
                    float *ZGeomData,
                    char *MFC)
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *XPos*

Pointer to double value specifying the X-position of the geometry.

YPos

Pointer to double value specifying the Y-position of the geometry.

ZPos

Pointer to double value specifying the Z-position of the geometry.

PosCoordMode

Pointer to integer value specifying the position coordinate system.

0=Grid

1=Frame

AttachToZone

Pointer to integer flag to signal that the geometry is "attached" to a zone.

Zone

Pointer to integer value specifying the number of the zone to attach to.

Color

Pointer to integer value specifying the color to assign to the geometry.

0=Black

1=Red

2=Green

3=Blue

4=Cyan

5=Yellow

6=Purple

7=White

8=Custom1

9=Custom2

10=Custom3

11=Custom4

12=Custom5

13=Custom6

14=Custom7

15=Custom8

FillColor

Pointer to integer value specifying the color used to fill the geometry. See *Color* above.

IsFilled

Pointer to integer flag to specify if geometry is to be filled.

GeomType

Pointer to integer value specifying the geometry type.

0=2DLineSegments

1=Rectangle

2=Square

3=Circle

4=Ellipse

5=3DLineSegments

LinePattern

Pointer to integer value specifying the line pattern.

0=Solid

1=Dashed

2=DashDot

3=Dotted

4=LongDash

5=DashDotDot

PatternLength

Pointer to double value specifying the pattern length in frame units.

LineThickness

Pointer to double value specifying the line thickness in frame units.

NumEllipsePts

Pointer to integer value specifying the number of points to use for circles and ellipses. The value must be greater than 0.

ArrowheadStyle

Pointer to integer value specifying the arrowhead style.

0=Plain

1=Filled

2=Hollow

ArrowheadAttachment

Pointer to integer value specifying where to attach arrowheads.

0=None

1=Beginning

2=End

3=Both

ArrowheadSize

Pointer to double value specifying the arrowhead size in frame units.

ArrowheadAngle

Pointer to double value specifying the arrowhead angle in degrees.

Scope

Pointer to integer value specifying the scope. 0=global, 1=local.

NumSegments

Pointer to integer value specifying the number of polyline segments.

NumSegPts

Array of integer values specifying the number of points in each of the *NumSegments* segments.

XGeomData

Array of floating-point values specifying the X-coordinates.

YGeomData

Array of floating-point values specifying the Y-coordinates.

ZGeomData

Array of floating-point values specifying the Z-coordinate.

MFC

Macro function command. Must be null terminated.

TECINI

Summary: Initializes the process of writing a binary data file. This must be called *first* before any other **TECIO** calls are made. You may write to multiple files by calling **TECINI** more than once. Each time **TECINI** is called, a new file is opened. Use **TECFIL** to switch between files.

FORTRAN Syntax:

```
  INTEGER*4 FUNCTION TECINI(Title,
 &                          Variables,
 &                          FName,
 &                          ScratchDir,
 &                          Debug,
 &                          VIsDouble)
  CHARACTER*(*) Title
  CHARACTER*(*) Variables
  CHARACTER*(*) FName
  CHARACTER*(*) ScratchDir
  INTEGER*4 Debug
  INTEGER*4 VIsDouble
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECINI(char *Title,
                char *Variables,
                char *FName,
                char *ScratchDir,
                INTEGER4 *Debug
                INTEGER4 *VIsDouble);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *Title*

Title of the data set. *Must be null terminated.*

Variables

List of variable names. If a comma appears in the string it will be used as the separator between variable names, otherwise a space is used. *Must be null terminated.*

FName

Name of the file to create. Must be null terminated.

ScratchDir

Name of the directory to put the scratch file. Must be null terminated.

Debug

Pointer to the integer flag for debugging. Set to 0 for no debugging or 1 to debug.

VIsDouble

Pointer to the integer flag for specifying whether field data generated in future calls to **TECDAT** are to be written in single or double precision. Set to 0 for single precision or 1 for double.

TECLAB

Summary: Write a set of custom labels to the data file.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECLAB(Labels)
 CHARACTER*(*)  Labels
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECLAB(char *Labels);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *Labels*

Character string of custom labels. Separate labels by a comma or space. For example, a set of custom labels for each day of the weeks is **Sun Mon Tue Wed Thu Fri Sat**.

TECNOD

Summary: Writes an array of node data to the binary data file. This is the connectivity list for finite element zones.

FORTRAN Syntax:

```
INTEGER*4 FUNCTION TECNOD(NData)
```

```
INTEGER*4 NData(T, M)
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECNOD(INTEGER4 *NData);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *NData*

Array of integers. This is the connectivity list, dimensioned (*T*, *M*) (*T* moving fastest), where *M* is the number of elements in the zone and *T* is set according to the following list:

ELEMENT TYPE	*T*
Triangle	3
Quadrilateral	4
Tetrahedral	4
Brick	8

TECTXT

Summary: Writes a text record to the data file.

FORTRAN Syntax:

```
    INTEGER*4 FUNCTION TECTXT(XPos,
  &                          YPos,
  &                          PosCoordMode,
  &                          AttachToZone,
  &                          Zone,
  &                          Font,
  &                          FontHeightUnits,
  &                          FontHeight,
  &                          BoxType,
  &                          BoxMargin,
  &                          BoxLineThickness,
  &                          BoxColor,
  &                          BoxFillColor,
  &                          Angle,
  &                          Anchor,
  &                          LineSpacing,
  &                          TextColor,
  &                          Scope,
  &                         Text,
  &                         MFC)
    DOUBLE PRECISION XPos
    DOUBLE PRECISION YPos
    INTEGER*4 PosCoordMode
    INTEGER*4 AttachToZone
    INTEGER*4 Zone
    INTEGER*4 Font
    INTEGER*4 FontHeightUnits
    DOUBLE PRECISION FontHeight
    INTEGER*4 BoxType
    DOUBLE PRECISION  BoxMargin
    DOUBLE PRECISION BoxLineThickness
    INTEGER*4 BoxColor
    INTEGER*4 BoxFillColor
    DOUBLE PRECISION Angle
    INTEGER*4 Anchor
    DOUBLE PRECISION LineSpacing
    INTEGER*4 TextColor
    INTEGER*4 Scope
    CHARACTER*(*) Text
    CHARACTER*(*) MFC
```

C Syntax:

```
#include TECIO.h

INTEGER4 TECTXT(double *XPos,
                double *YPos,
                INTEGER4 *PosCoordMode,
                INTEGER4 *AttachToZone,
                INTEGER4 *Zone,
                INTEGER4 *Font,
                INTEGER4 *FontHeightUnits,
                double *FontHeight,
                INTEGER4 *BoxType,
                double *BoxMargin,
                double *BoxLineThickness,
                INTEGER4 *BoxColor,
                INTEGER4 *BoxFillColor,
                double *Angle,
                INTEGER4 *Anchor,
                double *LineSpacing,
                INTEGER4 *TextColor,
                INTEGER4 *Scope,
                char *Text,
                char *MFC)
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *XPos*

Pointer to double value specifying the X-position of the geometry.

YPos

Pointer to double value specifying the Y-position of the geometry.

PosCoordMode

Pointer to integer value specifying the position coordinate system.

0=Grid 1=Frame

AttachToZone

Pointer to integer flag for to signal that the text is “attached” to a zone.

Zone

Pointer to integer value specifying the zone number to attach to.

Font

Pointer to integer value specifying the font.

0=Helvetica

1=Helvetica Bold

2=Greek

3=Math

4=User-Defined

5=Times

6=Times Italic

7=Times Bold

8=Times Italic Bold

9=Courier

10=Courier Bold

FontHeightUnits

Pointer to integer value specifying the font height units.

0=Grid

1=Frame

2=Point

FontHeight

Pointer to double value specifying the font height.

BoxType

Pointer to integer value specifying the box type.

0=None

1=Filled

2=Hollow

BoxMargin

Pointer to double value specifying the box margin (in frame units).

BoxLineThickness

Pointer to double value specifying the box line thickness (in frame units).

BoxColor

Pointer to integer value specifying the color to assign to the box.

0=Black

1=Red

2=Green

3=Blue

4=Cyan

5=Yellow

6=Purple

7=White

8=Custom1

9=Custom2

10=Custom3

11=Custom4

12=Custom5

13=Custom6

14=Custom7

15=Custom8

BoxFillColor

Pointer to integer value specifying the fill color to assign to the box. (See *BoxColor*)

Angle

Pointer to double value specifying the text angle in degrees.

Anchor

Pointer to integer value specifying where to anchor the text.

0=Left
1=Center
2=Right
3=MidLeft
4=MidCenter
5=MidRight
6=HeadLeft
7=HeadCenter
8=HeadRight

LineSpacing

Pointer to double value specifying the text line spacing.

TextColor

Pointer to integer value specifying the color to assign to the text. (See *BoxColor*)

Scope

Pointer to integer value specifying the scope.

0=Global
1=Local

Text

Character string representing text to display. Must be null terminated.

MFC

Macro function command. Must be null terminated.

TECZNE

Summary: Writes header information about the next zone to be added to the data file. After **TECZNE** is called, you must call **TECDAT** one or more times (and then call **TECNOD** if the data format is **FEBLOCK** or **FEPOINT**).

FORTRAN Syntax:

```
 INTEGER*4 FUNCTION TECZNE(ZoneTitle,
&                         L,
&                         M,
&                         N,
&                         ZoneFormat,
&                         DupList)
 CHARACTER*(*) ZoneTitle
```

```
          INTEGER*4 L
          INTEGER*4 M
          INTEGER*4 N
          CHARACTER*(*) ZoneFormat
       CHARACTER*(*)  DupList
```

C Syntax:

```
#include TECIO.h

lNTEGER4 TECZNE(char *ZoneTitle,
                INTEGER4 *L,
                INTEGER4 *M,
                INTEGER4 *N,
                char *ZoneFormat,
                char *DupList);
```

Return Value: `0` if successful, `-1` if unsuccessful.

Parameters: *ZoneTitle*

Title of the zone. *Must be null terminated.*

L, M, N

Pointers to integers specifying size of the zone. If the data is ordered (that is, zone format is **`BLOCK`** or **`POINT`**), then *L* is the I-dimension, *M* is the J-dimension, and *N* is the K-dimension. If the data is finite-element (that is, the zone format is **`FEBLOCK`** or **`FEPOINT`**), then *L* is the number of data points, *M* is the number of elements, and *N* is set according to the following chart:

ELEMENT TYPE	*N*
Triangle	0
Quadrilateral	1
Tetrahedron	2
Brick	3

ZoneFormat

Must be set to one of **`BLOCK`**, **`POINT`**, **`FEBLOCK`** or **`FEPOINT`**. Must be null terminated.

DupList

This parameter specifies a list of variables to duplicate from the preceding zone. For a complete explanation of the *DupList* parameter, see the *Tecplot User's Manual*. Must be null terminated.

The *DupList* parameter is a string of the following form:

```
"[n1,n2,...,nn][,FECONNECT]"
```

where *n1...nn* are the numbers of the variables to duplicate. If the zone is finite-element, you may optionally include **FECONNECT**, which will duplicate the connectivity list from the last zone.

Notes for using the *DupList* parameter:

- You cannot use the *DupList* parameter for the first zone, since in that case there is nothing to duplicate.
- If you use **FECONNECT**, you cannot call **TECNOD** for this zone, since **FECONNECT** specifies that the entire connectivity list from the previous zone will be duplicated.
- For finite-element zones, you can pass **"FECONNECT"** to duplicate only the connectivity list.
- You may pass either **NULL** or a 0 length string if you are not using this parameter.

Example: Duplicate variables 1 and 4 and the connectivity list. The *DupList* parameter must be set to:

```
"1,4,FECONNECT"//char(0)
```

11.7. Example Programs

This section lists example programs written both in FORTRAN and C which demonstrate the **TECIO** utility functions. These example programs can be found in the **util/tecio** directory below the Tecplot Home Directory. See the file **readme** in that directory for instructions on how to compile these examples. The first two examples use the old-style of **TECIO** functions, and the last two use the new (**100**) style.

11.7.1. Simple Example (FORTRAN)

```
C
C Simple example FORTRAN program to write a
C binary datafile for Tecplot. This example
C does the following:
C
C   1.  Open a data file called "t.plt"
C   2.  Assign values for X, Y and P
C   3.  Write out a zone dimensioned 4x5
C   4.  Close the data file.
C
C
```

```
      program test

      character*1 NULLCHR
      Integer*4   Debug,III,NPts,NElm

      Dimension X(4,5), Y(4,5), P(4,5)
      Integer*4 TecIni,TecDat,TecZne,TecNod,TecFil
      Integer*4 VIsDouble

      NULLCHR = CHAR(0)
      Debug   = 1
      VIsDouble = 0
      IMax    = 4
      JMax    = 5
      KMax    = 1
C
C... Open the file and write the Tecplot data file
C... header information.
C
      I = TecIni('SIMPLE DATASET'//NULLCHR,
     &           'X Y P'//NULLCHR,
     &           't.plt'//NULLCHR,
     &           '.'//NULLCHR,
     &           Debug,
     &           VIsDouble)

      Do 10 I  = 1,4
      Do 10 J  = 1,5
        X(I,J) = I
        Y(I,J) = J
        P(I,J) = I*J
   10 Continue
C
C... Write the zone header information.
C
      I = TecZne('Simple Zone'//NULLCHR,
     &           IMax,
     &           JMax,
     &           KMax,
     &           'BLOCK'//NULLCHR,
     &           CHAR(0))
C
C... Write out the field data.
C
      III = IMax*JMax
      I   = TecDat(III,X,0)
      I   = TecDat(III,Y,0)
      I   = TecDat(III,P,0)

      I = TecEnd()
      Stop
      End
```

11.7.2. Simple Example (C)

```
/*
 * Simple example C program to write a
 * binary data file for Tecplot.  This example
 * does the following:
 *
 *   1.  Open a datafile called "t.plt"
 *   2.  Assign values for X, Y and P
 *   3.  Write out a zone dimensioned 4x5
 *   4.  Close the data file.
 */

#include "TECIO.h"

main ()
{
  float X[5][4], Y[5][4], P[5][4];
  INTEGER4 Debug,I,J,III,DIsDouble,VIsDouble,IMax,JMax,KMax;

  Debug     = 1;
  VIsDouble = 0;
  DIsDouble = 0;
  IMax      = 4;
  JMax      = 5;
  KMax      = 1;
/*
 * Open the file and write the Tecplot data file
 * header information.
 */
  I = TECINI("SIMPLE DATASET",
             "X Y P",
             "t.plt",
             ".",
             &Debug,
             &VIsDouble);
  for (J = 0; J < 5; J++)
  for (I = 0; I < 4; I++)
    {
      X[J][I] = I+1;
      Y[J][I] = J+1;
      P[J][I] = (I+1)*(J+1);
    }
/*
 * Write the zone header information.
 */
  I = TECZNE("Simple Zone",
             &IMax,
             &JMax,
             &KMax,
             "BLOCK",
             NULL);
/*
 * Write out the field data.
 */
  III = IMax*JMax;
  I   = TECDAT(&III,&X[0][0],&DIsDouble);
  I   = TECDAT(&III,&Y[0][0],&DIsDouble);
  I   = TECDAT(&III,&P[0][0],&DIsDouble);

  I = TECEND();
}
```

11.7.3. Complex Example (FORTRAN)

```
C
C  Complex example FORTRAN program to write a
C  binary data file for Tecplot. This example
C  does the following:
C
C    1.  Open a data file called "field.plt."
C    2.  Open a data file called "line.plt."
C    3.  Assign values for X, Y and P. These will be used
C        in both the ordered and FE data files.
C    4.  Write out an ordered zone dimensioned 4 x 5 to "field.plt."
C    5.  Assign values for XL and YL arrays.
C    6.  Write out data for line plot to "line.plt."  Make the data
C        use double precision.
C    7.  Write out a finite element zone to "field.plt."
```

```
C    8.  Write out a text record to "field.plt."
C    9.  Write out a geometry (circle) record to "field.plt."
C   10.  Close file 1.
C   11.  Close file 2.
C
      Program ComplexTest

      REAL*4      X(4,5),  Y(4,5), P(4,5)
      REAL*8      XL(50), YL(50)
      REAL*4      XLDummy(1), YLDummy(1)
      EQUIVALENCE (XLDummy(1), XL(1))
      EQUIVALENCE (YLDummy(1), YL(1))
      INTEGER*4   Debug,I,J,K,L,III,NPts,NElm,DIsDouble,VIsDouble
      INTEGER*4   IMax,JMax,KMax,NM(4,12)
      REAL*8      XP, YP, ZP, FH, LineSpacing, PatternLength
      REAL*8      BoxMargin, BoxLineThickness, TextAngle
      INTEGER*4   AttachToZone, Zone, Scope, PositionCoordSys
      INTEGER*4   Clipping
      INTEGER*4   FontType, HeightUnits, Anchor, BoxType
      INTEGER*4   IsFilled, GeomType, LinePattern, NumEllipsePts
      INTEGER*4   BoxColor, BoxFillColor, TextColor, Color, FillColor
      INTEGER*4   ArrowheadStyle, ArrowheadAttachment, NumSegments
      INTEGER*4   NumSegPts(1)
      REAL*8      LineThickness, ArrowheadSize, ArrowheadAngle
      REAL*4      XGeomData(1), YGeomData(1), ZGeomData(1)
      CHARACTER*1 NULCHAR
      INTEGER*4   Zero
      POINTER     (NULLPTR, NULL)

      include "tecio.for"

      Debug     = 2
      VIsDouble = 0
      DIsDouble = 0
      NULCHAR   = CHAR(0)
      Zero      = 0
      NULLPTR   = 0
C
C Open field.plt and write the header information.
C
      I = TECINI100('DATASET WITH 1 ORDERED ZONE, 1 QUAD ZONE'//
     &              NULCHAR,
     &              'X Y P'//NULCHAR,
     &              'field.plt'//NULCHAR,
     &              '.'//NULCHAR,
     &                Debug,
     &                VIsDouble)
C
C  Open line.plt and write the header information.
C
      VIsDouble = 1
      I = TECINI100('DATASET WITH ONE I-ORDERED ZONE'//NULCHAR,
     &               'X Y'//NULCHAR,
     &               'line.plt'//NULCHAR,
     &              '.'//NULCHAR,
     &                Debug,
     &                VIsDouble)

C
C  Calculate values for the field variables.
C
      Do 10 J = 1,5
      Do 10 I = 1,4
          X(I,J) = I
          Y(I,J) = J
          P(I,J) = I*J
   10 Continue

C
C  Make sure writing to file #1.
C
      III = 1
      I = TECFIL100(III)

C
C  Write the zone header information for the ordered zone.
C
      IMax = 4
      JMax = 5
      KMax = 1
      I = TECZNE100('Ordered Zone'//NULCHAR,
     &                0, ! ZONETYPE
     &                IMax,
     &                JMax,
     &                KMax,
     &                0,     ! ICellMax
     &                0,     ! JCellMax
     &                0,     ! KCellMax
     &                1,     ! ISBLOCK
     &                0,     ! NumFaceConnections
     &               0,     ! FaceNeighborMode
     &               NULL,  ! ValueLocation
     &               NULL,  ! ShareVarFromZone
     &               0)     ! ShareConnectivityFromZone)

C
C  Write out the field data for the ordered zone.
C
      III = IMax*JMax
      I   = TECDAT100(III,X,DIsDouble)
      I   = TECDAT100(III,Y,DIsDouble)
      I   = TECDAT100(III,P,DIsDouble)

C
C  Calculate values for the I-ordered zone.
C

      Do 20 I = 1,50
         XL(I) = I
         YL(I) = sin(I/20.0)
   20 Continue
C
C  Switch to the 'line.plt' file (file number 2)
C  and write out the line plot data.
C
      III = 2
      I = TECFIL100(III)
C
C  Write the zone header information for the XY-data.
C
      IMax = 50
      JMax = 1
      KMax = 1
      I = TECZNE100('XY Line plot'//NULCHAR,
     &              0,
     &              IMax,
     &              JMax,
     &              KMax,
     &              0,
     &              0,
     &              0,
     &              1,
     &              0,
     &              0,
     &              NULL,
     &               NULL,
     &               0)
C
C  Write out the line plot.
C
      DIsDouble = 1
      III = IMax
      I   = TECDAT100(III,XLDummy,DIsDouble)
      I   = TECDAT100(III,YLDummy,DIsDouble)

C
C  Switch back to the field plot file and write out
C  the finite-element zone.
C
      III = 1
      I = TECFIL100(III)
C
C  Write the zone header information for the finite-element zone.
C
      NPts      = 20
      NElm      = 12
      KMax      = 1
      I = TECZNE100('Finite Zone'//NULCHAR,
     &               3,  ! FEQUADRILATERAL
     &               NPts,
     &               NElm,
     &               KMax,
     &               0,
     &               0,
     &               0,
     &               1,
     &               0,
     &               0,
     &               NULL,
     &               NULL,
     &               0)
C
C  Write out the field data for the finite-element zone.
C
      IMax      = 4
      JMax      = 5
      III       = IMax*JMax
      DIsDouble = 0
      I    = TECDAT100(III,X,DIsDouble)
      I    = TECDAT100(III,Y,DIsDouble)
      I    = TECDAT100(III,P,DIsDouble)

C
C  Calculate and then write out the connectivity list.
C  Note: The NM array references cells starting with
C        offset of 1.
C

      Do 30 I = 1,IMax-1
      Do 30 J = 1,JMax-1
          K = I+(J-1)*(IMax-1)
          L = I+(J-1)*IMax
          NM(1,K) = L
          NM(2,K) = L+1
          NM(3,K) = L+IMax+1
          NM(4,K) = L+IMax
   30 Continue

      I = TECNOD100(NM)

C
C  Prepare to write out text record. Text is positioned
C  at 50, 50 in frame units and has a height 5 frame units.
C
      XP                = 50
      YP                = 50
      FH                = 5
      Scope             = 1
      Clipping          = 0
      PositionCoordSys = 1
      FontType          = 1
      HeightUnits       = 1
      AttachToZone      = 0
      Zone              = 0
      BoxType           = 0
      BoxMargin         = 5.0
      BoxLineThickness = 0.5
      BoxColor          = 3
      BoxFillColor      = 7
      TextAngle         = 0.0
      Anchor            = 0
      LineSpacing       = 1.5
      TextColor         = 0

      III =  TECTXT100(XP,
     &                 YP,
     &                 0.0d0,
     &                 PositionCoordSys,
     &                 AttachToZone,
     &                 Zone,
     &                 FontType,
     &                 HeightUnits,
     &                 FH,
     &                 BoxType,
     &                 BoxMargin,
     &                 BoxLineThickness,
     &                 BoxColor,
     &                 BoxFillColor,
     &                 TextAngle,
     &                 Anchor,
     &                 LineSpacing,
     &                 TextColor,
     &                 Scope,
     &                 Clipping,
     &                'Hi Mom'//NULCHAR,
     &                ''//NULCHAR)

C
C  Prepare to write out geometry record (circle). Circle is
C  positioned at 25, 25 in frame units and has a radius of 30.
C  Circle is drawn using a dashed line pattern.
C


      XP                  = 25
      YP                  = 25
      ZP                  = 0.0
      IsFilled            = 0
      Color               = 0
      FillColor           = 7
      GeomType            = 2
      LinePattern         = 1
      LineThickness       = 0.3
      PatternLength       = 1
      NumEllipsePts       = 72
      ArrowheadStyle      = 0
      ArrowheadAttachment = 0
      ArrowheadSize       = 0.0
      ArrowheadAngle      = 15.0
      NumSegments         = 1
      NumSegPts(1)        = 1

      XGeomData(1) = 30
      YGeomData(1) = 0.0
      ZGeomData(1) = 0.0


      III =  TECGEO100(XP,
     &                 YP,
     &                 ZP,
     &                 PositionCoordSys,
     &                 AttachToZone,
     &                 Zone,
     &                 Color,
     &                 FillColor,
     &                 IsFilled,
     &                 GeomType,
     &                 LinePattern,
     &                 PatternLength,
     &                 LineThickness,
     &                 NumEllipsePts,
     &                 ArrowheadStyle,
     &                 ArrowheadAttachment,
     &                 ArrowheadSize,
     &                 ArrowheadAngle,
     &                 Scope,
     &                 Clipping,
     &                 NumSegments,
     &                 NumSegPts,
     &                 XGeomData,
     &                 YGeomData,
     &                 ZGeomData,
     &                 ''//NULCHAR)

C
C  Close out file 1.
C
      I = TECEND100()

C
C  Close out file 2.
C
      III = 2
      I = TECFIL100(III)
      I = TECEND100()
      STOP
      END
```

11.7.4. Complex Example (C)

```
/*
 * Complex example C program to write a
 * binary data file for Tecplot. This example
 * does the following:
 *
 *   1.  Open a data file called "field.plt."
 *   2.  Open a data file called "line.plt."
 *   3.  Assign values for X, Y and P. These will be used
 *       in both the ordered and finite-element data files.
 *   4.  Write out an ordered zone dimensioned 4 x 5 to "field.plt."
 *   5.  Assign values for XL and YL arrays.
 *   6.  Write out data for line plot to "line.plt." Make the data
 *       use double precision.
 *   7.  Write out a finite-element zone to "field.plt."
 *   8.  Write out a text record to "field.plt."
 *   9.  Write out a geometry (circle) record to "field.plt."
 *  10.  Close file 1.
 *  11.  Close file 2.
 */

#include <stdio.h>
#include <math.h>
#include "TECIO.h"

main ()
{
  float    X[5][4], Y[5][4], P[5][4];
  double   XL[50], YL[50];
  INTEGER4 Debug,I,J,K,L,III,NPts,NElm,DIsDouble,VIsDouble;
  INTEGER4 IMax,JMax,KMax;
  INTEGER4 ICellMax, JCellMax, KCellMax, ZoneType, Clipping;
  INTEGER4 IsBlock, NumFaceConnections;
  INTEGER4 FaceNeighborMode, ShareConnectivityFromZone;
  INTEGER4 NM[12][4];
  double   XP, YP, ZP, FH, LineSpacing, PatternLength;
  double   BoxMargin, BoxLineThickness, TextAngle;
  INTEGER4 AttachToZone, Zone, Scope, PositionCoordSys;
  INTEGER4 FontType, HeightUnits;
  INTEGER4 IsFilled, GeomType, LinePattern, NumEllipsePts;
  INTEGER4 Anchor, BoxType, BoxColor, BoxFillColor;
  INTEGER4 TextColor, Color, FillColor;
  INTEGER4 ArrowheadStyle, ArrowheadAttachment;
  INTEGER4 NumSegments, NumSegPts[1];
  double   LineThickness, ArrowheadSize, ArrowheadAngle;
  float    XGeomData[1], YGeomData[1], ZGeomData[1];

  Debug     = 2;
  VIsDouble = 0;
  DIsDouble = 0;
/*
 * Open order.plt and write the header information.
 */
  I = TECINI100("DATASET WITH ONE ORDERED ZONE AND ONE FE-QUAD ZONE",
                "X Y P",
                "field.plt",
                ".",
                &Debug,
                &VIsDouble);
/*
 * Open line.plt and write the header information.
 */
  VIsDouble = 1;
  I = TECINI100("DATASET WITH ONE I-ORDERED ZONE",
                "X Y",
                "line.plt",
                ".",
                &Debug,
                &VIsDouble);

/*
 * Calculate values for the field variables.
 */
  for (J = 0; J < 5; J++)
  for (I = 0; I < 4; I++)
    {
      X[J][I] = I+1;
      Y[J][I] = J+1;
      P[J][I] = (I+1)*(J+1);
    }

/*
 * Make sure writing to file #1.
 */
```

```
  III = 1;
  I = TECFIL100(&III);

/*
 * Write the zone header information for the ordered zone.
 */
  IMax      = 4;
  JMax      = 5;
  KMax      = 1;
  ICellMax  = 0;
  JCellMax  = 0;
  KCellMax  = 0;
  ZoneType  = 0;
  IsBlock   = 1;
  NumFaceConnections = 0;
  FaceNeighborMode   = 0;
  ShareConnectivityFromZone = 0;
  I = TECZNE100("Ordered Zone",
                &ZoneType,
                &IMax,
                &JMax,
                &KMax,
                &ICellMax,
                &JCellMax,
                &KCellMax,
                &IsBlock,
                &NumFaceConnections,
                &FaceNeighborMode,
                NULL,      /* ValueLocation */
                NULL,      /* ShareVarFromZone */
                &ShareConnectivityFromZone);
/*
 * Write out the field data for the ordered zone.
 */
  III = IMax*JMax;
  I   = TECDAT100(&III,&X[0][0],&DIsDouble);
  I   = TECDAT100(&III,&Y[0][0],&DIsDouble);
  I   = TECDAT100(&III,&P[0][0],&DIsDouble);

/*
 * Calculate values for the I-ordered zone.
 */

 for (I = 0; I < 50; I++)
   {
     XL[I] = I+1;
     YL[I] = sin((double)(I+1)/20.0);
   }
/*
 * Switch to the "line.plt" file (file number 2)
 * and write out the line plot data.
 */

  III = 2;
  I = TECFIL100(&III);

/*
 * Write the zone header information for the XY-data.
 */
  IMax = 50;
  JMax = 1;
  KMax = 1;
  I = TECZNE100("XY Line plot",
                &ZoneType,
                &IMax,
                &JMax,
                &KMax,
                &ICellMax,
                &JCellMax,
                &KCellMax,
                &IsBlock,
                &NumFaceConnections,
                &FaceNeighborMode,
                NULL,      /* ValueLocation */
                NULL,      /* ShareVarFromZone */
                &ShareConnectivityFromZone);
/*
 * Write out the line plot.
 */
  DIsDouble = 1;
  III = IMax;
  I   = TECDAT100(&III,(float *)&XL[0],&DIsDouble);
  I   = TECDAT100(&III,(float *)&YL[0],&DIsDouble);

/*
 * Switch back to the field plot file and write out
 * the finite-element zone.
 */
  III = 1;
  I = TECFIL100(&III);
```

```
/*
 * Write the zone header information for the finite-element zone.
 */
  ZoneType  = 3;  /* FEQuad */
  NPts      = 20; /* Number of points */
  NElm      = 12; /* Number of elements */
  KMax      = 0;  /* Unused */
  I = TECZNE100("Finite Zone",
                &ZoneType,
                &NPts,
                &NElm,
                &KMax,
                &ICellMax,
                &JCellMax,
                &KCellMax,
                &IsBlock,
                &NumFaceConnections,
                &FaceNeighborMode,
                NULL,      /* ValueLocation */
                NULL,      /* ShareVarFromZone */
                &ShareConnectivityFromZone);
/*
 * Write out the field data for the finite-element zone.
 */
  IMax      = 4;
  JMax      = 5;
  III       = IMax*JMax;
  DIsDouble = 0;
  I    = TECDAT100(&III,&X[0][0],&DIsDouble);
  I    = TECDAT100(&III,&Y[0][0],&DIsDouble);
  I    = TECDAT100(&III,&P[0][0],&DIsDouble);

/*
 * Calculate and then write out the connectivity list.
 * Note: The NM array references cells starting with
 *       offset of 1.
 */

  for (I = 1; I < IMax; I++)
  for (J = 1; J < JMax; J++)
    {
      K = I+(J-1)*(IMax-1);
      L = I+(J-1)*IMax;
      NM[K-1][0] = L;
      NM[K-1][1] = L+1;
```

```
        NM[K-1][2] = L+IMax+1;
        NM[K-1][3] = L+IMax;
      }

    I = TECNOD100((INTEGER4 *)NM);

  /*
   * Prepare to write out text record. Text is positioned
   * at 0.5, 0.5 in frame units and has a height
   * of 0.05 frame units.
   */
    XP                = 50.0;
    YP                = 50.0;
    ZP                = 0.0;
    FH                = 5.0;
    Scope             = 1; /* Local */
    Clipping          = 1; /* Clip to frame */
    PositionCoordSys  = 1; /* Frame */
    FontType          = 1; /* Helv Bold */
    HeightUnits       = 1; /* Frame */
    AttachToZone      = 0;
    Zone              = 0;
    BoxType           = 0; /* None */
    BoxMargin         = 5.0;
    BoxLineThickness  = 0.5;
    BoxColor          = 3;
    BoxFillColor      = 7;
    TextAngle         = 0.0;
    Anchor            = 0; /* Left */
    LineSpacing       = 1.0;
    TextColor         = 0; /* Black */

    III =  TECTXT100(&XP,
                     &YP,
                     &ZP,
                     &PositionCoordSys,
                     &AttachToZone,
                     &Zone,
                     &FontType,
                     &HeightUnits,
                     &FH,
                     &BoxType,
                     &BoxMargin,
                     &BoxLineThickness,
                     &BoxColor,
                     &BoxFillColor,
                     &TextAngle,
                     &Anchor,
                     &LineSpacing,
                     &TextColor,
                     &Scope,
                     &Clipping,
                     "Hi Mom",
                     "");

  /*
   * Prepare to write out geometry record (circle). Circle is
   * positioned at 25, 25 (in frame units) and has a radius of
   * 20 percent. Circle is drawn using a dashed line.
   */


    XP                  = 25.0;
    YP                  = 25.0;
    ZP                  = 0.0;
    IsFilled            = 0;
    Color               = 0;
    FillColor           = 7;
    GeomType            = 3; /* Circle */
    LinePattern         = 1; /* Dashed */
    LineThickness       = 0.3;
    PatternLength       = 1.5;
    NumEllipsePts       = 72;
    ArrowheadStyle      = 0;
    ArrowheadAttachment = 0;
    ArrowheadSize       = 0.0;
    ArrowheadAngle      = 15.0;
    NumSegments         = 1;
    NumSegPts[0]        = 1;

    XGeomData[0] = 20.0;
    YGeomData[0] = 0.0;
    ZGeomData[0] = 0.0;


    III =  TECGEO100(&XP,
                     &YP,
                     &ZP,
                     &PositionCoordSys,
                     &AttachToZone,
                             &Zone,
                             &Color,
                             &FillColor,
                             &IsFilled,
                             &GeomType,
                             &LinePattern,
                             &PatternLength,
                             &LineThickness,
                             &NumEllipsePts,
                             &ArrowheadStyle,
                             &ArrowheadAttachment,
                             &ArrowheadSize,
                             &ArrowheadAngle,
                             &Scope,
                             &Clipping,
                             &NumSegments,
                             NumSegPts,
                             &XGeomData[0],
                             &YGeomData[0],
                             &ZGeomData[0],
                             "");

  /*
   * Close out file 1.
   */
  I = TECEND100();

  /*
   * Close out file 2.
   */
  III = 2;
  I = TECFIL100(&III);
  I = TECEND100();
}
```

Index