Tecplot
Enjoy the View™

TECPLOT 360™
2006

Reference Manual

Tecplot, Inc. **Bellevue, WA** **2006**

Tecplot, Inc.
PO Box 52708
Bellevue, WA 98015-2708 U.S.A.
Tel: 1.800.763.7005 (within the U.S. or Canada), 00 1 (425)653-1200 (internationally)
email: sales@tecplot.com, support@tecplot.com
Questions, comments or concerns regarding this documentation: documentation@tecplot.com
For more information, visit http://www.tecplot.com

06-360-07-1

Rev 03/2006

CONTENTS

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 360 macro:

```
#!MC 1100
```

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 the macro file does not contain any function definitions, it will not be loaded into the Quick Macro Panel.

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

Writing Forward Compatible Macros

In order to ensure forward compatibility of your macro commands, please keep the following guidelines in mind. These guidelines will allow you to create macros that will work for years, on many machines and platforms.

1. **Begin your macro by opening a layout**.

   This will ensure that the final plot is consistent between versions of Tecplot (even if the default style settings for Tecplot have changed). Note: An alternative to using a layout is to load data and then paste a frame style file in each frame.

   If your macro will be used for more than one layout, you can ensure forward compatibility by:

   - Using the **$!PromptForFileName** command. This will allow the user to interactively specify the layout file.

     -or-

   - Launching Tecplot from the command line, specifying the layout and the macro: **tecplot mylayout.lay** *mydatafile* **mymacro.mcr**

2. Store associated files and graphics in the same folder as the macro file.

   If your macro loads files or inserts images without allowing the user to choose them, it is a good practice to store them in the same folder as the macro file that uses them. After recording, edit the macro, and replace the path to the file with the intrinsic macro variable |**macrofilepath**|.

   Example:

   $!Openlayout *"|macrofilepath|\Density.lpk"*

   This allows the macro to work without editing in any location as long as the entire folder of files was copied there.

3. Avoid using a $!Pick command in your macro.

   Changes to the aspect ratio can cause a recorded **$!Pick** command to fail when the macro is run on another machine or in another version of Tecplot.

- In a plot with multiple frames, don't use **$!Pick** to change the current frame. Instead, give each frame a meaningful name such as "Full View" and "Zoom Frame" in the layout. Then use the command:

  $!Framecontrol PopbyName Name = "Full View"

  to access the frame you want. This will also simplify later changes to the macro.

- If you must pick an item, make the pick as precise as possible. For example, clicking on the center, not the edge, of a zone or slice will increase the chances that the pick will be successful when the macro is replayed.

  When selecting text or geometries while recording a macro, click and drag in the widest possible area around the objects to select. The command will be recorded as

  ```
  $!PICK ADDALLINRECT
    SELECTTEXT = YES
    X1 = 1.56075949367
    X2 = 3.97088607595
    Y1 = 2.29556962025
    Y2 = 3.91582278481
  ```

  The x and y ranges can be expanded if needed.

4. Use plenty of comments in your macro.

Chapter 4

Debugging macros

In general, the best way to debug a macro is to use the Macro Viewer, and find which command is causing the problem.

Here are some tips for specific problems:

Problem: The macro was created with a previous version of Tecplot to make the plot needed. With a newer version of Tecplot, the macro will run without error, but the plot looks different.

Solution: Run the macro with the old version of Tecplot, then save a frame style to a file. Begin your macro by loading the data, then pasting the frame style file from a file. This will ensure that the final plot will be consistent from one version of Tecplot to the next, even if the default style settings for Tecplot have been changed.

Problem: The macro gives you errors such as “File does not exist” or “Cannot open file”, but you can locate the file.

Solution: Copy the file to the same folder as the macro file that uses the file. Edit the macro, and replace the path to the file with the intrinsic macro variable `|macrofilepath|`.

Example: `$!Openlayout "|macrofilepath|\Density.lpk"`

This allows the macro to work without editing in any location as long as the entire folder of files was copied there.

Problem: Running the macro causes unusual error messages, such as: “No objects to cut or the objects selected not allowed to be cut”, or “Not allowed to adjust zones or mappings when the mouse mode is set to SELECTOR.” When you run the macro in the Macro Viewer, you see that the problem occurs with when a **`$!Pick`** command is run.

Solution: Avoid using a **$!Pick** command in your macro. Changes to the aspect ratio can cause a recorded **$!Pick** command to fail when the macro is run on another machine or in another version of Tecplot.

To fix the problem in an existing macro, follow these steps to make the coordinates more precise:

1. Run the macro on the machine where the error message is generated.
2. Via the macro viewer or editor, identify the preceding **$!PICK ADD** or similar select type pick command. Note the X,Y coordinates of the command. A good way to do this is:
   a. Run the macro until you get the “No Objects to Shift” error message.
   b. Click Ok on the dialog.
   c. Bring up the macro viewer: **File>Macro>Viewer**
   d. Find the nearest **$!Pick ADD** command above the current command and put a break point on that command.
   e. Press “Reset” to reset the macro and then run the macro.
   NOTE: If the problem only occurs when running in batch mode then try to determine the macro command by examining the batch.log file. Insert a **$!Pause** command in your macro just before the **$!Pick Add** command that precedes the offending command. Now run Tecplot interactively from the macro viewer. You can then see the line number where you need to put the break.
3. Back in Tecplot, select the zoom tool.
4. Hold the shift key down and notice that the running coordinates in the lower right corner now show “PX = xxxxx PY = yyyyyy". xxxxxx and yyyyyy are the paper coordinates of the hot spot of the zoom tool. (If you see X and Y for grid coordinates, or FX and FY for frame coordinates, you need to hold down the Shift key. Pick commands always use paper coordinates.)
5. Move the zoom tool until xxxxx and yyyyy are close to the coordinates noted in step 2.
6. Note where the pick occurred. It is likely the pick occurred some distance away from the actual edge of the object to pick. Move the zoom tool to a “better” location for the pick and note the coordinates.
7. Edit the macro file and replace the old X,Y pick coordinates with those determined in step 6.

Chapter 5 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 360, this line is **#!MC 1100.**

A Tecplot 360 macro file has the form:

#!MC 1100
<*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 6 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|
```

6 - 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.
|AXISMAXY|	Maximum value of current Y-axis range.
|AXISMAXZ|	Maximum value of current Z-axis range.

Variables	Notes
|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.
|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.

Variables	Notes
|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.
|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*.

Variables	Notes
|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.
|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.
|SOLUTIONTIME|	Retrieve Tecplot's current solution time.
|SOLUTIONTIME[[ACTIVEOFFSET=]nnn]|	Retrieve the solution time of zone nnn. If ACTIVEOFFSET= is used, the integer value indicates the first zone associated with the nnn'th active field map. |SOLUTIONTIME[5]| would retrieve the solution time of the 5th zone. |SOLUTIONTIME[ACTIVEOFFSET=3]| would retrieve the solution time of the first zone in the 3rd active field map.
|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.

Variables	Notes
|STREAMTYPE|	The streamtrace type such as “Surface Line”, or “Surface Ribbon”
|TECHOME|	Path to the Tecplot home directory.
|TECPLOTVERSION|	Currently returns 110.
|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.

6 - 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.

6- 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.

6- 2.2 Example 2

To create a string from an environment variable:

```
$!VARSET |AUTHOR| = "Author: |$LOGNAME|"
```

6 - 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**.

6 - 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
```

6 - 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.

6 - 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|"

6 - 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 1100
$!MACROFUNCTIONNAME = "3D Rotation Animation"
$!EXPORTSETUP EXPORTFORMAT = AVI
$!EXPORTSETUP IMAGEWIDTH = 546
$!EXPORTSETUP EXPORTFNAME = "|1|AxisRotation.avi"
$!EXPORTSTART
$!LOOP |2|
  ANGLE = 3
  ROTATEORIGINLOCATION = DEFINEDORIGIN
$!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")
```

6 - 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 7

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.

$!ACTIVEFIELDMAPS

A SetValue command that changes the set of active field maps (thus changing the active zones) considered for plotting.

$!ACTIVELINEMAPS

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

$!ADDMACROPANELTITLE

Add a title to the Quick Macro Panel.

$!ADDONCOMMAND

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.

$!ALTERDATA

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 <*zonelist*> parameter specifies the set of zones on which to operate, where zonelist is a list of zones or zone ranges separated by a comma (","). Zone ranges are separated by a hyphen ("-"). If <*zonelist*> is omitted, all zones are affected. NOTE: the values for the <*zonelist*> parameter must be enclosed in square brackets (i.e. **$!ALTERDATA [1,3]** to apply **ALTERDATA** to zones 1 and 3).

$!ANIMATECONTOURLEVELS

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.

$!ANIMATEIJKBLANKING

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.

$!ANIMATEIJKPLANES

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.

$!ANIMATEISOSURFACES

The macro command **$!ANIMATEISOSURFACES** produces an animation of a series of iso-surfaces beginning with the iso-surface defined by **STARTVALUE** and ending with the iso-surface defined by **ENDVALUE**. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

$!ANIMATELINEMAPS

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

$!ANIMATESLICES

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

$!ANIMATESTREAM

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.

$!ANIMATETIME

Produce an animation of transient data. To create an AVI or RM file, add $!EXPORTSETUP commands before this command.

$!ANIMATEZONES

Produce an animation showing one zone at a time. To create an AVI or RM file,

add $!EXPORTSETUP commands before this command. NOTE: this command will not work if the current frame contains a transient data set.

$!ATTACHDATASET

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.

$!ATTACHGEOM

Attach a geometry to the current frame.

$!ATTACHTEXT

Attach text to the current frame.

$!BASICCOLOR

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

$!BASICSIZE

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.

$!BLANKING

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

$!BRANCHCONNECTIVITY

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

$!BRANCHFIELDDATAVAR

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

$!BREAK

Jump out of the current `$!LOOP-ENDLOOP` or `$!WHILE-$!ENDWHILE`.

$!COLORMAPCONTROL [<groupnumber>] [Required-Control Option]

The different commands in the `COLORMAPCONTROL` compound function family are described separately in the following sections. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

$!COLORMAPCONTROL [<groupnumber>] REDISTRIBUTECONTROLPOINTS

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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

$!COLORMAPCONTROL [<groupnumber>] COPYSTANDARD

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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

$!COLORMAPCONTROL [<groupnumber>] RESETTOFACTORY

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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

$!COMPATIBILITY

Allow datasharing access and setting, without warning.

$!CONTINUE

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

$!CONTOURLABELS [Required-Control Option]

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

$!CONTOURLABELS ADD

Add contour labels to your plot.

$!CONTOURLABELS DELETEALL

Delete all currently defined contour labels.

$!CONTOURLEVELS [Required-Control Option]

The different commands in the `CONTOURLEVELS` compound function family are described separately in the following sections.

$!CONTOURLEVELS ADD

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

$!CONTOURLEVELS DELETENEAREST

Delete the contour level whose value is nearest the value supplied in the `RANGEMIN` parameter.

$!CONTOURLEVELS DELETERANGE

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

$!CONTOURLEVELS NEW

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

$!CONTOURLEVELS RESET

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

$!CONTOURLEVELS RESETTONICE

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.

$!CREATECIRCULARZONE

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

$!CREATECONTOURLINEZONES

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.

$!CREATEFEBOUNDARY

Zone edges for finite element data cannot be turned on or off using the edge 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.

$!CREATEFESURFACEFROMIORDERED

A FE-Surface zone can be generated from two or more I-Ordered zones. To get the best possible output, it is recommended 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.

$!CREATEISOZONES

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.

$!CREATELINEMAP

Create a new Line-mapping.

$!CREATEMIRRORZONES

Create new zones that are mirror images of the source zones

$!CREATENEWFRAME

Creates a new frame.

$!CREATERECTANGULARZONE

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 ini-

tialized to zero.

$!CREATESIMPLEZONE

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.

$!CREATESLICEZONEFROMPLANE

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

$!CREATESLICEZONES

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

$!CREATESTREAMZONES

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.

$!DATASETUP

A SetValue command that sets miscellaneous parameters related to data.

$!DEFAULTGEOM

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.

$!DEFAULTTEXT

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.

$!DELAY

Delay Tecplot execution for *<integer>* seconds.

$!DELETEAUXDATA

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

$!DELETELINEMAPS

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

$!DELETEVARS

Delete one or more variables.

$!DELETEZONES

Delete one or more zones.

$!DOUBLEBUFFER [Required-Control Option]

The different commands in the DOUBLEBUFFER compound function family are described separately in the following sections.

$!DOUBLEBUFFER OFF

Turn off double buffering; use this command once at the end of a sequence of

using the double buffer.

$!DOUBLEBUFFER ON

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.

$!DOUBLEBUFFER SWAP

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

$!DRAWGRAPHICS

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.

$!DROPDIALOG

Drop a Tecplot interface dialog. This command is mainly useful for the Tecplot demo. To launch a dialog use $!LAUNCHDIALOG.

$!DUPLICATELINEMAP

Copy attributes from an existing Line-mapping to another.

$!DUPLICATEZONE

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.

$!ELSE

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

$!ELSEIF

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

$!EXPORT

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.)

$!EXPORTCANCEL

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

$!EXPORTFINISH

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.

$!EXPORTNEXTFRAME

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.)

$!EXPORTSETUP

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 gen-

erating output intended for printers and plotters.

$!EXPORTSTART

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.)

$!EXTRACTFROMGEOM

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.

$!EXTRACTFROMPOLYLINE

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*>.

$!FIELDLAYERS

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

$!FIELDMAP

A SetValue command that assigns zone attributes for field plots. The <*set*> parameter immediately following the `$!FIELDMAP` 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*>.

$!FILECONFIG

A SetValue command that sets file path information in Tecplot.

$!FONTADJUST

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

$!FRAMECONTROL [Required-Control Option]

The different commands in the `FRAMECONTROL` compound function family are described separately in the following sections.

$!FRAMECONTROL DELETETOP

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.)

$!FRAMECONTROL FITALLTOPAPER

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

$!FRAMECONTROL POP

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

$!FRAMECONTROL POPATPOSITION

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

$!FRAMECONTROL POPBYNAME

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

$!FRAMECONTROL PUSH

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

$!FRAMECONTROL PUSHBYNAME

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

$!FRAMECONTROL PUSHTOP

Push the top (active) frame to the bottom.

$!FRAMELAYOUT

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.

$!FRAMENAME

Set the name for the current frame.

$!FRAMESETUP

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

$!GETAUXDATA

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

$!GETCONNECTIVITYREFCOUNT

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

$!GETCURFRAMENAME

Query Tecplot for the name of the current frame. The <*macrovar*> represents the macro vari-

able to receive the results.

$!GETFIELDVALUE

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.

$!GETFIELDVALUEREFCOUNT

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

$!GETNODEINDEX

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.

$!GETVARLOCATION

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

$!GETVARNUMBYNAME

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.

$!GETZONETYPE

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

$!GLOBALCOLORMAP

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.

$!GLOBALCONTOUR

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.

$!GLOBALEDGE

A SetValue command that sets attributes which sets the minimum crease angle for edges.

$!GLOBALFRAME

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

$!GLOBALLINEPLOT

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

$!GLOBALPOLAR

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

$!GLOBALRGB

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

$!GLOBALSCATTER

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

$!GLOBALTHREED

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

$!GLOBALTHREEDVECTOR

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

$!GLOBALTIME

A SetValue command for frames (2D and 3D ONLY). Different frames can have different values of **$!GLOBALTIME**

$!GLOBALTWODVECTOR

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

$!IF...$!ENDIF

Conditionally process macro commands.

$!INCLUDEMACRO

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.

$!INTERFACE

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

$!INVERSEDISTINTERPOLATE

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

$!ISOSURFACEATTRIBUTES

A SetValue command which changes attributes associated with iso-surfaces. The optional group parameter can range from 1-4 and defaults to 1 when absent.

$!ISOSURFACELAYERS

$!KRIG

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

$!LAUNCHDIALOG

Launch a Tecplot interface dialog; This command is mainly useful for the Tecplot demo.

$!LIMITS

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.

$!LINEARINTERPOLATE

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.

$!LINEMAP

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>*.

$!LINEPLOTLAYERS

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

$!LINKCOLORMAPS

Set to true to tie all colormaps together.

$!LINKING

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

$!LOADADDON

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.

$!LOADCOLORMAP

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

$!LOOP...$!ENDLOOP

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.

$!MACROFUNCTION...$!ENDMACROFUNCTION

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.

$!NEWLAYOUT

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

$!OPENLAYOUT

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

$!PAPER

A SetValue command that sets the paper characteristics.

$!PAUSE

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.

$!PICK [Required-Control Option]

The different commands in the `PICK` compound function family are described separately in the following sections.

$!PICK ADD

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

$!PICK ADDALL

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

$!PICK ADDALLINRECT

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).

$!PICK CLEAR

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

$!PICK COPY

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

$!PICK CUT

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

$!PICK EDIT

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.

$!PICK MAGNIFY

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.

$!PICK PASTE

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

$!PICK POP

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.

$!PICK PUSH

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.

$!PICK SETMOUSEMODE

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).

$!PICK SHIFT

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`.)

$!PLOTTYPE

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

$!POLARAXIS

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

$!POLARTORECTANGULAR

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

$!POLARVIEW

Sets the viewing style for polar plots in a layout.

$!PRINT

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

$!PRINTSETUP

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.

$!PROMPTFORFILENAME

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).

$!PROMPTFORTEXTSTRING

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*>.

$!PROMPTFORYESNO

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.

$!PROPAGATELINKING

Link multiple frames, ether within frame or between frames.

$!PUBLISH

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

$!QUIT

Terminate the execution of the Tecplot program.

$!RAWCOLORMAP

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.

$!READDATASET

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

$!READSTYLESHEET

Read in a stylesheet file. The *<string>* is the name of the file to read.

$!REDRAW

Redraw the current frame.

$!REDRAWALL

Redraw all frames.

$!REMOVEVAR

Remove a user-defined macro variable. This frees up space so another user-defined macro variable can be defined.

$!RENAMEDATASETVAR

Rename a data set variable in Tecplot.

$!RENAMEDATASETZONE

Rename a data set zone in Tecplot.

$!RESET3DAXES

Reset the ranges on the 3-D axes.

$!RESET3DORIGIN

Reposition the rotation origin in 3-D to be at the specified location.

$!RESET3DSCALEFACTORS

Recalculate the scale factors for the 3-D axes. Aspect ratio limits are taken into account.

$!RESETVECTORLENGTH

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`.

$!ROTATE2DDATA

Rotate field data in 2-D about any point.

$!ROTATE3DVIEW

Do a 3-D rotation about a given axis. The <*rotateaxis*> must be supplied.

$!RUNMACROFUNCTION

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.

$!SAVELAYOUT

Save the current layout to a file. You must supply the file name.

$!SET3DEYEDISTANCE

Sets the distance from the viewer to the plane of the current center of rotation.

$!SETAUXDATA

Add Auxiliary 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 by one or more underscore, period, letter, or digit characters.

$!SETDATASETTITLE

Set the title for the current data set.

$!SETFIELDVALUE

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.

$!SETFRAMEBACKGROUNDCOLOR

Sets the frame background to the specified color and surveys all basic color assignments in Tecplot, converting the all basic colors using the following rules to achieve the best contrast:

$!SETSTYLEBASE

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.

$!SHARECONNECTIVITY

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.

$!SHAREFIELDDATAVAR

Allows sharing of the specified variable from the source 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.

$!SHIFTLINEMAPSTOBOTTOM

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.

$!SHIFTLINEMAPSTOTOP

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.

$!SHOWMOUSEPOINTER

The mouse icon may be deactivated within a macro to enhance the on-screen animation. It must be reactivated before exiting the macro.

!SKETCHAXIS

A SetValue command that assigns attributes for axes in a sketch mode frame. Axes are rarely used in sketch frames.

$!SLICEATTRIBUTES

A SetValue command that changes global attributes associated with slices.

$!SLICELAYERS

$!SMOOTH

Smooth data (reduce the spikes) for selected variables in selected zones.

$!STREAMATTRIBUTES

A SetValue command that changes global attributes associated with streamtraces.

$!STREAMTRACE [Required-Control Option]

The different commands in the STREAMTRACE compound function family are described separately in the following sections.

$!STREAMTRACE ADD

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.

$!STREAMTRACE DELETEALL

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.

$!STREAMTRACE DELETERANGE

Delete a range of streamtraces. Streamtraces are numbered sequentially in the order they were created.

$!STREAMTRACE RESETDELTATIME

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.

$!STREAMTRACE SETTERMINATIONLINE

Set the position of the termination line for streamtraces.

$!STREAMTRACELAYERS

$!SYSTEM

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.

$!THREEDAXIS

A SetValue command that assigns attributes for axes in a 3-D frame.

$!THREEDVIEW

A SetValue command that changes global attributes associated with the 3-D

view.

$!TRANSFORMCOORDINATES

Transforms all points in one or more zones from one coordinate system to another.

$!TRIANGULATE

Create a new zone by forming triangles from data points in existing zones.

$!TWODAXIS

A SetValue command that assigns attributes for axes in a 2-D frame.

$!VARSET

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.

$!VIEW [Required-Control Option]

The different commands in the `VIEW` compound function family are described separately in the following sections.

$!VIEW AXISFIT

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.

$!VIEW AXISMAKECURRENTAXISVALUESNICE

Reset the axis-line label values such that all currently displayed values are set to have the smallest number of significant digits possible.

$!VIEW AXISNICEFIT

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.

$!VIEW CENTER

Center the data within the axis grid area.

$!VIEW COPY

Copy the current view to the view paste buffer. See also `$!VIEW PASTE`.

$!VIEW DATAFIT

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.

$!VIEW FIT

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.

$!VIEW LAST

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.

$!VIEW MAKECURRENTVIEWNICE

Shifts axis to make axis-line values nice without changing the extents of the window. Only works in Sketch/XY/2D.

$!VIEW NICEFIT

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.

$!VIEW PASTE

Retrieve the view from the view paste buffer and assign it to the current frame.

$!VIEW PUSH

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.

$!VIEW RESETTOENTIRECIRCLE

Reset the Theta-R Axis to initial settings. For Polar plots only.

$!VIEW SETMAGNIFICATION

Set the magnification for the data being plotted. A magnification of 1 will size the plot so it can fit within the grid area.

$!VIEW TRANSLATE

Shift the data being plotted in the X- and/or Y-direction. The amount translated is in frame units.

$!VIEW ZOOM

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.

$!WHILE...$!ENDWHILE

Continue to execute a set of commands until a conditional expression is false.

$!WORKSPACEVIEW [Required-Control Option]

The different commands in the `WORKSPACEVIEW` compound function family are described separately in the following sections.

$!WORKSPACEVIEW FITALLFRAMES

Change the view in the workspace so all frames are fit just inside the edges of the workspace.

$!WORKSPACEVIEW FITPAPER

Change the view in the workspace so the entire paper is fit just inside the edges of the workspace.

$!WORKSPACEVIEW FITSELECTEDFRAMES

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.

$!WORKSPACEVIEW LASTVIEW

Return to the previous workspace view.

$!WORKSPACEVIEW MAXIMIZE

Temporarily expand the work area as large as possible. The maximized work area occupies the entire Tecplot process window.

$!WORKSPACEVIEW TRANSLATE

Shift the view of the workspace. This has no effect on the local view within any frame in your layout.

$!WORKSPACEVIEW UNMAXIMIZE

Returns the workspace to its normal size after it has been expanded after $!WORKSPACE MAXIMIZE has been used.

$!WORKSPACEVIEW ZOOM

Change the view into the work area. This has no effect on the local view within any frame in your layout.

$!WRITECOLORMAP

Write the current color map to a file. The *<string>* is the name of the file to write to.

$!WRITECURVEINFO

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.

$!WRITEDATASET

Write the data set attached to the current frame to a file. The *<string>* is the name of the file to write to.

$!WRITESTYLESHEET

Write the style for the current frame to a file. The *<string>* is the name of the file to write to.

$!XYLINEAXIS

A SetValue command that assigns attributes for axes in an XY Line plot.

ANIMATESTREAKLINES

Animates previously calculated streaklines to the screen or to a file.

ATTACHINTEGRATIONRESULTS

Attach the text results of the previous integration as a text field in the current frame.

CALCPARTICLEPATH

Calculate particle paths or streaklines, starting from existing Tecplot streamtraces.

CALCTURBULENCEFUNCTION

Calculate a turbulence-related function from two variables in the current data set. Add the result to the data set as a new variable using the function's name, or overwrite the variable if it already exists.

CALCULATE

Calculate a Tecplot variable using the specified function and add it to the current data set. If the variable already exists in the current data set, it will be recalculated.

CALCULATEACCURACY

Calculate the order accuracy of the solution contained in the listed zones. Optionally, plot the overall accuracy versus grid spacing and plot the accuracy at each grid node.

DISPLAYBOUNDARIES

Displays boundaries corresponding to a geometry and boundaries specification without actually setting the geometry and boundaries. This macro is generally not useful for those writing macro files, but is recorded when the user clicks the Display Boundaries button in the Geometry and Boundaries dialog in order to duplicate the actions of Tecplot that happen in response to that action. See "SETGEOMETRYANDBOUNDARIES" on page 241 for a description of the parameters for this macro.

EXTRACTFLOWFEATURE

Extract and display shock surfaces, vortex cores, or separation and attachment lines. Shock surfaces are displayed as isosurfaces of a new variable, ShockSurface, while vortex cores and separation and attachment lines are displayed as new zones.

EXTRAPOLATESOLUTION

Perform Richardson extrapolation to estimate the true solution from three input solutions on grids of successively finer resolution. Two new zones are added to the current data set. The first contains the extrapolated solution, while the second contains the estimated error.

INTEGRATE

Perform an integration over the specified zones. If **<*set*>** is not specified, the integration will be performed over all zones. If **PLOTAS** is set to **TRUE**, the integration results will be plotted in a new frame.

SAVEINTEGRATIONRESULTS

Saves the most recently calculated integration results to a text file.

SETFIELDVARIABLES

Identifies variables in your data, such as velocity, pressure and temperature, for

use in analysis.

SETFLUIDPROPERTIES

Set the fluid properties for use by other commands.

SETGEOMETRYANDBOUNDARIES

Specify whether the data represent an axisymmetric flow solution (2D Cartesian plots only), whether adjacent zones should be considered to be connected at coincident faces, and specify zone boundaries and their corresponding boundary conditions.

SETREFERENCEVALUES

Specify the reference (free-stream) properties of the solution, identify two variables in the current data set for use with other commands.

SETUNSTEADYFLOWOPTIONS

Identifies time levels for unsteady flow, or specifies that the solution is steady-state. If the flow is unsteady, the solution time levels are specified in the **RAWDATA** section. The first line of the **RAWDATA** section must consist of a single integer indicating the number of solution time levels. This must be followed by the time levels themselves. Each time level must be on a separate line and must consist of a floating-point number (the solution time), as well as one or more integers (the zone numbers for that solution time).

<<anchorpos>>

Assign attributes for positioning of objects.

<<areastyle>>

Change settings for the axis grid area.

<<axisdetail>>

Assign attributes for axes.

<<axisline>>

Assign attributes for axis lines.

<<axistitle>>

Assign attributes for titles.

<<basicsizelist>>

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.

<<colormapcontrolpoints>>

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.

<<colormapoverride>>

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.

<<continuouscolor>>

Change settings for continuous color.

<<dialogplacement>>

Describes the placement for a dialog.

<<gridarea>>

<<gridlinedetail>>

Change settings for axis gridlines.

<<ijk>>

Set an I-, J- or K-index.

<<indexrange>>

Set an index range.

<<numberformat>>

Set the format used to draw a number.

<<papersize>>

Change dimensions or hardclip offsets for **`LETTER`**, **`DOUBLE`**, **`A3`**, **`A4`**, **`CUSTOM1`** and **`CUSTOM2`** paper sizes.

<<precisegrid>>

<<rect>>

Change settings for a rectangle. The rectangle is defined using two points (X1,Y1) and (X2,Y2).

<<refscatsymbol>>

Set the attributes for the reference scatter symbol.

<<renderconfig>>

Set the attributes for OpenGL rendering.

<<rgb>>

Set a color value by assigning values to its red, green, and blue components.

<<shademap>>

Map colors on the screen to shades of gray for monochrome hardcopy output.

<<symbolshape>>

Set a symbol shape. Symbols can be a geometric shape (circle, square, and so forth) or an ASCII character.

<<textbox>>

Change settings for the optional box around a text label.

<<textshape>>

Change settings related to text font and character height.

<<ticklabeldetail>>

Change settings for the text used to label axis tick marks.

<<tickmarkdetail>>

Assign attributes for axis tick marks.

<<volumeobjectstoplot>>

Specifies what volume objects are to be displayed.

<<xy>>

Change settings for an (X,Y) position.

<<xyz>>

Change settings for an (X, Y, Z) triplet.

<<zebrashade>>

Change zebra shading attributes.

Chapter 8 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 10 "Parameter Subcommands".

$!ACTIVEFIELDMAPS

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

Description: A SetValue command that changes the set of active field maps (thus changing the active zones) considered for plotting.

Examples:

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

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

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

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

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

```
$!ACTIVEFIELDMAPS - = [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 *[zonelist]*
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 <*zonelist*> parameter specifies the set of zones on which to operate, where *zonelist* is a list of zones or zone ranges separated by a comma ("**,**"). Zone ranges are separated by a hyphen ("**-**"). If <*zonelist*> is omitted, all zones are affected. NOTE: the values for the <*zonelist*> parameter must be enclosed in square brackets (i.e. **$!ALTERDATA [1,3]** to apply **ALTERDATA** to zones 1 and 3).

Required Parameters:

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

Optional Parameters:

Parameter	Syntax	Default	Notes
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).
IRANGE **{** MIN MAX SKIP **}**	 = *<integer>* = *<integer>* = *<integer>*	 1 0 1	See the note, Range Parameters, for information on specifying range index values.
JRANGE **{** MIN MAX SKIP **}**	 = *<integer>* = *<integer>* = *<integer>*	 1 0 1	See the note, Range Parameters, for information on specifying range index values.

Parameter	Syntax	Default	Notes
KRANGE { MIN MAX SKIP }	 = <integer> = <integer> = <integer>	 1 0 1	See the note, Range Parameters, for information on specifying range index values.
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*. 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 [1,3]
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}
```

Example 4: By following a variable reference with brackets "**[**" and "**]**" you may designate a specific zone from which to get the variable value. For example:

```
V3 = V3 -V3[1]
X = (X[1] + X[2] + X[3]) / 3
{TempAdj} = {Temp}[7] - {Adj}
V7 = V1[19] - 2*C[21] + {R/T}[18]
```

The zone number must be a positive integer constant less than or equal to the number of zones. The zone designated must have the same structure (finite-element, I-, IJ-, or IJK-ordered) and dimensions (number of nodes and so forth)

$!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	Default	Notes
START	= *<integer>*		Starting contour level number to animate.
END	= *<integer>*		Ending contour level number to animate.

Optional Parameters:

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

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`.
`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`.

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

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
```

$!ANIMATEISOSURFACES

Syntax:

```
$!ANIMATEISOSURFACES
    STARTVALUE = <double>
    ENDVALUE = <double>
  [optional parameters]
```

Description: The macro command **$!ANIMATEISOSURFACES** produces an animation of a series of iso-surfaces beginning with the iso-surface defined by **STARTVALUE** and ending with the iso-surface defined by **ENDVALUE**. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Required Parameters:

Parameter	Syntax	Default	Notes
ENDVALUE	= *<integer>*		ENDVALUE is the value of the contour variable for the last iso-surface in the animation.
NUMSTEPS	= *<integer>*	2	Number of iso-surfaces to distribute between the start and end iso-surfaces values.
STARTVALUE	= *<integer>*		STARTVALUE is the value of the contour variable for the first iso-surface in the animation.

Optional Parameters:

Parameter	Syntax	Default	Notes
CREATEMOVIEFILE	= *<boolean>*	FALSE	If TRUE, must be preceded by $!EXPORTSETUP commands.
GROUP	= *<integer>*	1	values 1- 4
LIMITSCREENSPEED	= *<boolean>*	No	
MAXSCREENSPEED	= *<double>*		You may need to reduce the value to correlate with the speed of your computer.

Note: *Go To*, *Loop*, *Bounce*, *Forward*, and *Backward* are only used by the interface. *Forward* and *Backward* can be simulated using appropriate values **STARTVALUE** and **ENDVALUE**. If **ENDVALUE < STARTVALUE**, the animation goes 'backward'. If **ENDVALUE > STARTVALUE**, the animation goes 'forward'. *Goto* can be simulated if **ENDVALUE == STARTVALUE**, i.e. the animation goes 'one step'. *Loop* and *Bounce* can be accomplished by calling the file multiple times.

Note: When recording, the macro recorded contains exactly the animation done in the interface. So if you bounce three times through the data, you will record three sets of forward and backwards commands. Similarly, if you use the "one step" options a lot, you will record a lot of individual macro commands. If you interrupt part way through an animation, you will record a partial animation macro of those steps you did animate through.

Example: The following example creates an animation of iso-surfaces:

```
$!ANIMATEISOSURFACES
   STARTVALUE = 1
   ENDVALUE = 30
   NUMSTEPS = 30
```

$!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** [Group]
START = *<integer>*
END = *<integer>*
[optional parameters]

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

Required Parameters:

Parameter	Syntax	Default	Notes
START	= *<integer>*		START and END are measured in steps based on NUMSLICES between the slice group's start slice value (at step=1) and end slice values (at step = NumSlices).
END	= *<integer>*		START and END are measured in steps based on NUMSLICES between the slice group's start slice value (at step=1) and end slice values (at step = NumSlices).
NUMSLICES	= *<integer>*	2	Number of slices to distribute between the start and end slice locations as defined by **START and END** in **$!SLICEATTRIBUTES**.

Optional Parameters:

Parameter	Syntax	Default	Notes
GROUP	= *<integer>*	1	values 1- 4
CREATEMOVIEFILE	= *<boolean>*	**FALSE**	If TRUE, must be preceded by $!EXPORTSETUP commands.
LIMITSCREENSPEED	= *<boolean>*		
MAXSCREENSPEED	= *<double>*		

Note: *Go To*, *Loop*, *Bounce*, *Forward*s, and *Backwards* are only used by the interface. They can be simulated by using the correct **STARTVALUE** and **ENDVALUE**. If **ENDVALUE** < **STARTVALUE**, the animation goes 'backwards'. If **ENDVALUE** > **STARTVALUE**, the animation goes 'forwards'. And if **ENDVALUE** == **STARTVALUE**, the animation goes 'one step'. *Loop* and *Bounce* can be accomplished with multiple calls.

Note: When recording, the macro recorded contains exactly the animation done in the interface. So if you bounce three times through the data, you will record three sets of forward and backwards commands. Similarly, if you use the "one step" options a lot, you will record a lot of individual macro commands. If you interrupt part way through an animation, you will record a partial animation macro of those steps you did animate through.

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

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

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

$!ANIMATETIME

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

Description: Produce an animation of transient data. To create an AVI or RM file, add **$!EXPORTSETUP** commands before this command.

Optional Parameters:

Parameter	Syntax	Default	Notes
CREATEMOVIEFILE	**=** *<boolean>*	FALSE	If TRUE, must be preceded by $!EXPORTSETUP commands.
ENDTIME	= *<double>*	The last timestep as defined by the currently active strands	If the SolutionTime entered does not exist, the nearest SolutionTime less than the entered time is used.
LIMITSCREENSPEED	**=** *<boolean>*	FALSE	
MAXSCREENSPEED	= *<double>*	12	only works if LIMITSCREENSPEED is TRUE
SKIP	= *<integer>*	1	
STARTTIME	= *<double>*	first time step as defined by the currently active strands	If the SolutionTime entered does not exist, the nearest SolutionTime less than the entered time is used.

Note: Go To, Loop, Bounce, Forwards, and Backwards are only used by the interface. These operations can be simulated by using the correct STARTTIME and ENDTIME. If ENDTIME < STARTTIME, the animation goes 'backwards'. If ENDTIME > STARTTIME, the animation goes 'forwards'. And if ENDTIME == STARTTIME, the animation goes 'one step'.

Loop and Bounce can be accomplished with multiple calls to $!ANIMATETIME.

$!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. NOTE: this command will not work if the current frame contains a transient data set.

Required Parameters:

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

.**Optional Parameters:**

Parameter	Syntax	Default	Notes
ZONEANIMATIONMODE	= *[STEPBYNUMBER, GROUPSTEPBYNUMBER, STEPBYTIME]*	STEPBYNUMBER	
CREATEMOVIEFILE	= *<boolean>*	FALSE	If TRUE, must be preceded by $!EXPORTSETUP commands.
SKIP	= *<integer>*	1	Zone skip.

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:

$!ATTACHGEOM

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
ANCHORPOS	<<anchorpos>>		This assigns the anchor position of the geometry.
ARROWHEADANGLE	= *<dexp>*	12	Set the angle for arrowheads (in degrees).
ARROWHEADATTACHMENT	<arrowheadattachment>	NONE	
ARROWHEADSIZE	= *<dexp>*	5%	Set the arrowhead size in Y-frame units (0-100).
ARROWHEADSTYLE	<arrowheadstyle>	PLAIN	
ATTACHTOZONE	= *<boolean>*	FALSE	If **TRUE**, must include **ZONE.**
CLIPPING	= *<clipping>*	CLIPTTOVIEWPORT	
COLOR	= *<color>*	BLACK	
DATATYPE	= *<fielddatatype>*	FLOAT	
DRAWORDER	= *<draworder>*	AFTERDATA	
FILLCOLOR	= *<color>*	WHITE	
GEOMTYPE	= *<geomtype>*	LINESEGS	
IMAGEFILENAME	= *<string>*		
ISFILLED	= *<boolean>*		
LINEPATTERN	= *<linepattern>*	SOLID	
LINETHICKNESS	= *<dexp>*	0.1%	Set the line thickness in Y-frame units (0-100).
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.
MAINTAINASPECTRATIO	= *<boolean>*	TRUE	

Parameter	Syntax	Default	Notes
NUMELLIPSEPTS	= *<integer>*	72	Numbers of points to use when drawing ellipses and circles.
PATTERNLENGTH	= *<dexp>*	2%	Set the pattern length in Y-frame units (0-100).
POSITIONCOORDSYS	= *<coordsys>*	GRID	
RESIZEFILTER	= *<resizefilter>*		
SCOPE	= *<scope>*	LOCAL	Set the scope to **GLOBAL** to draw this geometry in all "like" frames.
TEXTUREFILTER		CUBIC	
ZONE	= *<integer>*	1	This is only used if **ATTACHTOZONE** = **TRUE**. This geometry is disabled if the zone assigned here is inactive.

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
ANCHOR	= *<textanchor>*	LEFT	Specifies what part of the text to anchor to the frame.
ANCHORPOS	<<anchorpos>>		This assigns the anchor position for the text. Units are dependent on POSITIONCOORDSYS.
ANGLE	= *<dexp>*	0.0	Text angle (in degrees).
ATTACHTOZONE	= *<boolean>*	FALSE	If TRUE, must include ZONE.
BOX { BOXTYPE COLOR FILLCOLOR LINETHICKNESS MARGIN }	 = *<boxtype>* = *<color>* = *<color>* = *<dexp>* = *<dexp>*	 NONE BLACK WHITE 0.1% 20	 The margin is the space between the text and box. The margin is measured in terms of the percentage of the text height.
CLIPPING	= *<clipping>*	CLIPTOVIEWPORT	
COLOR	= *<color>*	BLACK	
LINESPACING	= *<dexp>*	1.0	Line spacing to use if text contains multiple lines.

Parameter	Syntax	Default	Notes
MACROFUNCTIONCOMMAND	= *<string>*	NULL	Set the macro command to execute when you hover over the geometry and press Ctrl-right-click.
POSITIONCOORDSYS	= *<coordsys>*	FRAME	values = FRAME, GRID or GRID3D
TEXTSHAPE { FONT HEIGHT SIZEUNITS }	 = *<font>* = *<dexp>* = *<sizeunits>*	 HELVBOLD 14 POINT	 The following combinations of SIZEUNITS and POSITIONCOORDSYS are allowed: FRAME/FRAME, POINT
SCOPE	= *<scope>*	LOCAL	Set the scope to GLOBAL to include this text in all “like” frames.
ZONE	= *<integer>*	1	This is only used if ATTACHZONE = TRUE. This text is disabled if the zone assigned here is inactive.

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
ANCHORPOS {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
}
ANCHORPOS {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>>	
`BLUE`	<<rgb>>	
`CUSTOM1...CUSTOM56`	<<rgb>>	
`CYAN`	<<rgb>>	
`GREEN`	<<rgb>>	
`PURPLE`	<<rgb>>	
`RED`	<<rgb>>	
`WHITE`	<<rgb>>	
`YELLOW`	<<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
`ARROWHEADSIZES`	<<basicsizelist>>	
`FRAMETEXTSIZES`	<<basicsizelist>>	
`LINEPATLENGTHS`	<<basicsizelist>>	

Parameter	Syntax	Notes
LINETHICKNESSES	<<basicsizelist>>	
POINTTEXTSIZES	<<basicsizelist>>	
SYMBOLSIZES	<<basicsizelist>>	
TICKLENGTHS	<<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
DEPTH {		
INCLUDE	= <*boolean*>	If TRUE, draws only those portions at the plot with depth values within the FROMFRONT and FROMBACK limits.
FROMFRONT	= <*double*>	FROMFRONT and FROMBACK are expressed as percentages of the overall 3-D depth.
FROMBACK	= <*double*>	FROMFRONT and FROMBACK are expressed as percentages of the overall 3-D depth.
}		
IJK {		
INCLUDE		
IJKBLANKMODE	<*op*> <*boolean*>	
IMINFRACT	= <*ijkblankmode*>	Minimum and maximum fractions are in terms of percentages (0-100). Zero represents an index of one and 100 the maximum index.
JMINFRACT	<*op*> <*dexp*>	
KMINFRACT	<*op*> <*dexp*>	
IMAXFRACT	<*op*> <*dexp*>	
JMAXFRACT	<*op*> <*dexp*>	
KMAXFRACT	<*op*> <*dexp*>	
ZONE	= <*integer*>	Only one zone can be assigned to use IJK-blanking.
}		
VALUE		

Parameter	Syntax	Notes
{		
BLANKENTIRECELL	= <boolean>	Set to **FALSE** to get precision-blanking.
CONSTRAINT nnn	*nnn* = <integer>	Use <integer> to specify which constraint to modify.
{		
COLOR	= <color>	
CONSTRAINTOP2MODE	= <constrainintop2mode>	
INCLUDE	= <boolean>	
LINEPATTERN	= <linepattern>	
LINETHICKNESS	= <double>	
PATTERNLENGTH	= <double>	
RELOP	= <valueblankrelop>	
SHOW	= <boolean>	
VALUECUTOFF	= <double>	
VARA	= <integer>	
VARB	= <integer>	
}		
INCLUDE	= <boolean>	Set to **FALSE** to turn off all value-blanking.
VALUEBLANKCELLMODE	= <valueblankcellmode>	
}		

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
VAR	= <integer>	
ZONE	= <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
```

$!COLORMAPCONTROL [<*groupnumber*>] *[Required-Control Option]*

Description: The different commands in the **COLORMAPCONTROL** compound function family are described separately in the following sections. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

The **COLORMAPCONTROL** compound functions are:

$!COLORMAPCONTROL [<groupnumber>**]**
REDISTRIBUTECONTROLPOINTS

$!COLORMAPCONTROL [<groupnumber>] COPYSTANDARD

$!COLORMAPCONTROL [<groupnumber>] RESETTOFACTORY

$!COLORMAPCONTROL [<*groupnumber*>] REDISTRIBUTECONTROLPOINTS

Syntax: **$!COLORMAPCONTROL [**<groupnumber>**] 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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

Example: `$!COLORMAPCONTROL REDISTRIBUTECONTROLPOINTS`

$!COLORMAPCONTROL [<*groupnumber*>] COPYSTANDARD

Syntax: `$!COLORMAPCONTROL [<groupnumber>] 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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

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 [<*groupnumber*>] RESETTOFACTORY

Syntax: `$!COLORMAPCONTROL [<groupnumber>] 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. Group number is an optional parameter ranging from 1 to 4, which defaults to 1 when omitted.

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

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

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` [group]
[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 edges for finite element data cannot be turned on or off using the edge 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.

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 recommended 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
`HEIGHT`	`=` *<dexp>*	8	Units are in inches.
`XYPOS`			
`{`			
X	`=` *<dexp>*	1.0	X-position (inches) relative to the left edge of the paper.
Y	`=` *<dexp>*	0.25	Y-position (inches) relative to the top edge of the paper.
`}`			
`WIDTH`	`=` *<dexp>*	9	Units are in inches.

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 instructions.
`YVAR`	= <integer>	Auto	Only needed when processing journal instructions.
`ZVAR`	= <integer>	Auto	Only needed when processing journal instructions.
`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 12 "Raw Data" on page 285 for details.

Optional Parameter:

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

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:

```
$!GLOBALCONTOUR VAR = 4
$!SLICEATTRIBUTES ENDPOSITION {X = 1}
$!SLICEATTRIBUTES STARTPOSITION {X = 6}
$!SLICEATTRIBUTES NUMITERMEDIATESLICES = 6
$!SLICEATTRIBUTES SHOWBEGINENDSLICE = YES
$!SLICEATTRIBUTES SHOWINTERMEDIATESLICES = YES
$!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	Default	Notes
COMMANDLINE { AutoStrandDataFiles }	= *<boolean>*	True	: This option allows you to auto-strand data files in Tecplot360. This can be set to FALSE or commented-out of the configuration file (tecplot.cfg) to retain the Tecplot 10 compatibility
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 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	Default	Notes
ANCHORPOS	<<xyz>>		
ARROWHEADANGLE	<op> <dexp>		
ARROWHEADATTACHMENT	<arrowheadattachment>		
ARROWHEADSIZE	<op> <dexp>		
ARROWHEADSTYLE	<arrowheadstyle>		
ATTACHTOZONE	= <boolean>		
COLOR	= <color>		
DATATYPE	= <fielddatatype>		
DRAWORDER	= <draworder>	AFTERDATA	
DRAWORDER	= <draworder>	AFTERDATA	
FILLCOLOR	= <color>		
ISFILLED	= <boolean>		
LINEPATTERN	= <linepattern>		
LINETHICKNESS	<op> <dexp>		
MACROFUNCTIONCOMMAND	= <string>		Set the macro command to execute when you hover over the geometry and press Ctrl-right-click.
MAINTAINASPECTRATIO	= <boolean>	TRUE	
NUMELLIPSEPTS	<op> <integer>		
PATTERNLENGTH	<op> <dexp>		
PIXELASPECTRATIO	= <double>	0	A value of 0 allows Tecplot to select the aspect ratio.Use only if your circles or squares due to the aspect ratio of your monitor.
POSITIONCOORDSYS	= <coordsys>		
SCOPE	= <scope>		
ZONE	= <integer>		

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

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/var/dataset/frame/linemap]*
[optional parameters]

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

Required Parameters:

Parameter	Syntax	Notes
AUXDATALOCATION	= *[zone/var/dataset/ frame/linemap]*	

Optional Parameters:

Parameter	Syntax	Notes
NAME	= *<string>*	
NUM	= *<integer>*	
VAR	= *<integer>*	
ZONE	= *<integer>*	Only required if **AUXDATALOCATION** = zone

Example: Delete the selected Auxiliary 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:

```
$!DELETEVARS [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. 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
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.
SOURCEMAP	= <integer>	Line-mapping from which to copy.

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 MAX SKIP **}**	 = <integer> = <integer> = <integer>	 1 0 1	See notes on index ranges for $!ALTERDATA action command.
JRANGE **{** MIN MAX SKIP **}**	 = <integer> = <integer> = <integer>	 1 0 1	See notes on index ranges for $!ALTERDATA action command.
KRANGE **{** MIN MAX SKIP **}**	 = <integer> = <integer> = <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	Default	Notes
ANIMATIONSPEED	= *<double>*		Applies to AVI only. Sets the animation speed in frames per second.
CONVERTTO256COLORS	= *<boolean>*		Used for TIFF, BMP, and PNG formats.
EXPORTFNAME	= *<string>*		
EXPORTFORMAT	= *<exportformat>*		
EXPORTREGION	= <bitdumpregion>		
FLAHSHCOMPRESSIONITYPE	= *<compressiontype>*		
FLASHIMAGETYPE	= *<imagetype>*		
IMAGEWIDTH	*<op>* *<integer>*		
JPEGENCODING	= *STANDARD or PROGRESSIVE*		
PRINTRENDERTYPE	= *<printrendertype>*	VECTOR	
QUALITY	= *<integer>*		Range is from 1-100
SUNRASTERFORMAT	= *<sunrasterformat>*		Only applies if EXPORTFORMAT is SUNRASTER.
SUPERSAMPLEFACTOR	= *<integer>*	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.
TIFFBYTEORDER	= *<tiffbyteorder>*		
USEMULTIPLECOLORTABLES	= *<boolean>*		Applies to AVI and Raster Metafile only.
USESUPERSAMPLEANTIALIASING	= *<boolean>*	FALSE	

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
EXPORTREGION = CURRENTFRAME
$!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.
`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.
`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.

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
EXTRACTLINEPOINTSONLY	= *<boolean>*	FALSE	If FALSE, must include NUMPTS.
EXTRACTTHROUGHVOLUME	= *<boolean>*	FALSE	If TRUE, data is extracted from XYZ-coordinates in the polyline. If FALSE, data is extracted from the surface.
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.
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.

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
```

$!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
SHOWCONTOUR	= <boolean>	
SHOWEDGE	= <boolean>	
SHOWISOSURFACES	= <boolean>	
SHOWMESH	= <boolean>	
SHOWSCATTER	= <boolean>	
SHOWSHADE	= <boolean>	
SHOWSLICES	= <boolean>	
SHOWVECTOR	= <boolean>	Vector variables must be defined. See $!GLOBALTWODVECTOR or $!GLOBALTHREEDVECTOR.
TWODDRAWORDER	= <twoddraworder>	
USELIGHTINGEFFECT	= <boolean>	
USETRANSLUCENCY	= <boolean>	

Example: Turn on the scatter layer:

```
$!FIELDLAYERS
SHOWSCATTER = YESP
```

$!FIELDMAP

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

Description: A SetValue command that assigns zone attributes for field plots. The <set> parameter immediately following the **`$!FIELDMAP`** 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
CONTOUR {		

Parameter	Syntax	Notes
COLOR	= <color>	
CONTOURTYPE	= <meshtype>	
FLOODCOLORING	= <contourcoloring>	
LINECONTOURGROUP	= <integer>	
LINEPATTERN	= <linepattern>	
LINETHICKNESS	<op> <dexp>	
PATTERNLENGTH	<op> <dexp>	
SHOW	= <boolean>	
USELIGHTINGEFFECT	= <boolean>	
}		
EDGELAYER		
{		
COLOR	= <color>	
EDGETYPE	= <edgetype>	
IEDGE	= <borderlocation>	Applies for IJ-, IK-, and IJK-ordered zones.
JEDGE	= <borderlocation>	Applies for IJ-, IK-, and IJK-ordered zones.
KEDGE	= <borderlocation>	Applies for IJ-, IK-, and IJK-ordered zones.
LINETHICKNESS	= <dexp>	
SHOW	= <boolean>	
USEBLANKING	= <boolean>	
}		
EFFECTS		
{		
LIGHTINGEFFECT	= <lightingeffect>	
SURFACETRANSLUCENCY	<translucency>	SURFACETRANSLUCENCY range is one to 99.
USETRANSLUCENCY	= <boolean>	
USEVALUEBLANKING	= <boolean>	Set to TRUE to include value blanking in the specified zones
}		
MESH		
{		
COLOR	= <color>	
LINEPATTERN	= <linepattern>	
LINETHICKNESS	<op> <dexp>	
MESHTYPE	= <meshtype>	
PATTERNLENGTH	<op> <dexp>	
SHOW	= <boolean>	
}		
POINTS		
{		
IJKSKIP	<<ijk>>	Limits the number of vectors or scatter symbols drawn.
POINTSTOPLOT	<pointstoplot>	
}		
SCATTER		
{		
COLOR	= <color>	
FILLCOLOR	= <color>	

Parameter	Syntax	Notes
FILLMODE	= <*fillmode*>	
FRAMESIZE	<*op*> <*dexp*>	Size of symbols when SIZEBYVARIABLE is FALSE.
LINETHICKNESS	<*op*> <*dexp*>	
SHOW	= <*boolean*>	
SIZEBYVARIABLE	= <*boolean*>	Scatter sizing variable must be defined before this can be set to TRUE. See the $!GLOBALSCATTER command.
SYMBOLSHAPE	<<symbolshape>>	
}		
SHADE		
{		
COLOR	= <*color*>	
SHOW	= <*boolean*>	
USELIGHTINGEFFECT	= <*boolean*>	
}		
SURFACES		
{		
IRANGE	<<indexrange>>	
JRANGE	<<indexrange>>	
KRANGE	<<indexrange>>	
SURFACESTOPLOT	= <*surfacestoplot*>	
}		
VECTOR		
{		
ARROWHEADSTYLE	<*arrowheadstyle*>	
COLOR	= <*color*>	
ISTANGENT	= <*boolean*>	
LINEPATTERN	= <*linepattern*>	
LINETHICKNESS	= <*dexp*>	
PATTERNLENGTH	= <*dexp*>	
SHOW	= <*boolean*>	
VECTORTYPE	= <*vectortype*>	
}		
VOLUMEMODE		VOLUMEMODE applies to volume zones, with the exception that POINTSTOPLOT also applies to finite-element surface zones.
{		
VOLUMEOBJECTSTOPLOT	<<volumeobjectstoplot>>	
}		
ZONEGROUP	= <*integer*>	Assign a group number to the supplied set of zones.

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

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

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

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

$!FILECONFIG

Syntax: `$!FILECONFIG`
[optional parameters]

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

Optional Parameters:

Parameter	Syntax	Default	Notes
ADDZONESTOEXISTINGSTRANDS	= *<boolean>*	False	If TRUE, Tecplot will add the zones from the appended data to any existing strands in the dataset. If FALSE, Tecplot will append the strands from the new data to any existing strands in the dataset.
ASSIGNSTRANDID	= *<boolean>*	True	If TRUE, Tecplot will assign strand ID's to zones (if time is supplied for the zones but not strand ID's). If FALSE, Tecplot will not associate these zones with any strands.
DATAFILEVARLOADMODE	= *<varloadmode>*	BYNAME	Set the default loading mode for variables. To get Tecplot Version 7.0 behavior, use **BYPOSITION**.
DOAUTOFNAMEEXTENSION	= *<boolean>*		
DOAUTOFNAMEEXTENSIONWARNING	= *<boolean>*		If TRUE a warning is displayed when attempting to save with an extension other than the default extension.
FNAMEFILTER **{**			
COLORMAPFILE	= *<string>*		Default extension for color map files.
EQUATIONFILE	= *<string>*		Default extension for equation files.
IMPORTIMAGEFILE	= *<string>*		Default extension for image files.
INPUTDATAFILE	= *<string>*		Default extension for Tecplot input data files.
INPUTLAYOUTFILE	= *<string>*		Default extension for loading layout files.
MACROFILE	= *<string>*		Default extension for macro files.

Parameter	Syntax	Default	Notes
OUTPUTASCIIDATAFILE	= <string>		Default extension for ASCII output data files.
OUTPUTBINARYDATAFILE	= <string>		Default extension for binary output data files.
OUTPUTLAYOUTFILE	= <string>		Default extension for saving linked layout files.
OUTPUTLAYOUTPACKAGEFILE	= <string>		Default extension for saving layout package files.
STYLEFILE	= <string>		Default extension for style files.
}			
LAYOUTCONFIG			
{			
INCLUDEDATA	= <boolean>	FALSE	If TRUE, layout packages are the default format for layouts
INCLUDEPREVIEW	= <boolean>	FALSE	If TRUE, preview images are saved with layout packages.
USERELATIVEPATHS	= <boolean>	TRUE	If **TRUE**, files will be referenced using relative paths in layout files.
}			
LOADONDEMAND			
{			
ALLOW	= <boolean>		If TRUE, Tecplot will use it's load-on-demand features for loading and unloading variables.
DATASTORESTATEGY	= <dataloadstrategy>	AUTO	Set the data store strategy for load-on-demand. If set to **AUTO**, Tecplot will use store large allocations in the temporary directory and use memory mapped I/O to read and write to the regions when possible otherwise it will use the memory heap (usually this provides better performance for large data). If set to **HEAP** Tecplot will not use the temporary directory for large allocations (this option is usually slower when working with large data).

Parameter	Syntax	Default	Notes
UNLOADSTRATEGY }	= *<unloadstrategy>*	AUTO	Set the unload strategy for load-on-demand. If set to **AUTO** Tecplot will unload unused variables when the amount needed RAM begins to reach the maximum amount of RAM. If set to **NEVERUNLOAD** Tecplot will load variables on demand but will never attempt to unload them even if it is running low on memory. If set to **MINIMIZEMEMORYUSE** Tecplot will aggressively unload variables as soon as they are not needed regardless of the amount of memory available or in use.
MEMORYMAPPEDIOTHRESHOLD	= *<integer>*		Minimum size (in bytes) of data before Tecplot will consider using memory mapped I/O. If the value is zero Tecplot will let the operating system decide which is usually the best choice.
TEMPFILEPATH	= *<string>*		Set the directory where you want Tecplot to store its temporary files.
USEMEMORYMAPPEDIO	= *<boolean>*	TRUE	If TRUE, Tecplot will use the operating system's memory mapped I/O facility to read data faster whenever possible

File Name Filters: Valid characters 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
BOLDFACTOR	*<op>* *<double>*	Thickness of bold characters relative to normal.
INTERCHARSPACING	*<op>* *<integer>*	Increase or decrease inter-character spacing. Units are in pixels on the screen.
STROKEFONTLINETHICKNESS	*<op>* *<double>*	Thickness (in frame units) of lines used to draw stroke fonts.
SUBSUPFRACTION	*<op>* *<double>*	Size of subscript and superscript characters relative to the font height.

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
```

$!FRAMELAYOUT

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
BACKGROUNDCOLOR	= <color>	Only applies if ISTRANSPARENT = FALSE.
BORDERTHICKNESS	<op> <dexp>	Value is in Y-frame units.
HEADERCOLOR	= <color>	Only applies if SHOWHEADER = TRUE.
HEADERFONT	= <font>	
HEIGHT	<op> <dexp>	Value is in inches.
ISTRANSPARENT	= <boolean>	
SHOWBORDER	= <boolean>	
SHOWHEADER	= <boolean>	
WIDTH	<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.
FITINITIALFRAMETOWORKSPACE	*<boolean>*	If set to FALSE, the new layout is shown as in V10 with the entire paper fit to the work area. If set to TRUE, the new layout is shown with the current frame fit to the work area. This command changes the behavior of Tecplot as it first appears during a session and as it appears after a new layout command. It has no effect on the current plot, but it can be used in a macro to set the value for future new plots. It is typically found in the *tecplot.cfg* file.
INITIAL3DSCALE	*<op>* *<dexp>*	Initial scale for 3-D plots.
NUMSTREAMRAKEPOINTS	*<op>* *<integer>*	Number of points to place along streamtrace rakes.
RODRIBBONDEFLEN	*<op>* *<dexp>*	Default width (in frame units) of a streamtrace or ribbon
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.
VECTMINLEN	*<op>* *<dexp>*	Minimum length in centimeters. Vectors shorter than this length are not drawn.

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/var/dataset/ frame/linemap]*	
NAME	= *<string>*	Name of existing auxiliary data

Optional Parameters:

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

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

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
VAR	= <*integer*>	
ZONE	= <*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
```

$!GLOBALCOLORMAP

Syntax: `$!GLOABLCOLORMAP [`*<groupnumbers>*`]`
[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	Default	Notes
CONTOURCOLORMAP	*<colormap>*		
GRAYSCALE	<<colormapcontrolpoints>>		
GROUPNUMBER	*<integer>*	1	Group number must be between 1 and 4.
LGRAINBOW	<<colormapcontrolpoints>>		
MODERN	<<colormapcontrolpoints>>		
SMRAINBOW	<<colormapcontrolpoints>>		
TWOCOLOR	<<colormapcontrolpoints>>		
USERDEFINED	<<colormapcontrolpoints>>		
USERDEFINED NUMCONTROLPOINTS	= *<integer>*		

Example: Make the third control point for the small rainbow color map for the 4th Color Map group to be positioned 0.44 of the way across the color map. Set the leading and trailing RGB red value to 90:

```
$!GLOBALCOLORMAP 4
SMRAINBOW
{
 CONTROLPOINT 3
 {
 COLORMAPFRACTION = 0.44
 LEADRGB
 {R = 90}
 TRAILRGB
 {R = 90}
}
}
```

$!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	Default	Notes
CONTOURLINESTYLE **{** CONTOURLINEMODE LINESKIP PATTERNLENGTH **}**	 **=** *<contourlinemode>* *<op>* *<integer>* *<op>* *<dexp>*		This is used to assign a special line pattern scheme for contour line plots.
COLORCUTOFF **{** INCLUDEMAX INCLUDEMIN RANGEMAX RANGEMIN **}**	 = *<boolean>* = *<boolean>* *<op>* *<dexp>* *<op>* *<dexp>*		
COLORMAPFILTER **{** COLORMAPCYCLES COLORMAPDISTRIBUTION COLORMAPOVERRIDE COLORMAPOVERRIDEACTIVE CONTINUOUSCOLOR REVERSECOLORMAP USEFASTSPPROXCONTINUOUSFLOOD ZEBRA **}**	 *<op>* *<integer>* *<colormapdistribution>* *<integer>* *<<colormapoverride>>* **=** *<boolean>* <<continuouscolor>> **=** *<boolean>* **=** *<boolean>* <<zebrashade>>	 FALSE	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.

Parameter	Syntax	Default	Notes
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.
LABELS **{**			
ALIGNAUTOLABELS	= <boolean>		If TRUE, automatic labels are aligned with the contour lines, otherwise they are horizontal.
AUTOLABELSPACING	<op> <dexp>		
AUTOLEVELSKIP	<op> <integer>		Value is in Y-frame units.
COLOR	= <color>		
FILLCOLOR	= <color>		
GENERATEAUTOLABELS	= <boolean>		If TRUE, automatic labels are repositioned on each redraw.
ISFILLED	= <boolean>		
LABELWITHVALUE	= <boolean>		If TRUE, automatic labels show the contour value otherwise they show the contour level number.
MARGIN	<op> <dexp>		
NUMFORMAT	<<numberformat>>		
SHOW	= <boolean>		
TEXTSHAPE	<<textshape>>		Not allowed to change size units parameter.
}			
LEGEND **{**			
ANCHORALIGNMENT	<anchoralignment>		
AUTORESIZE	= <boolean>		
AUTOSIZEMAXLIMIT	= <double>		

Parameter	Syntax	Default	Notes
BOX	<<textbox>>		
HEADERTEXTSHAPE	<<textshape>>		
INCLUDECUTOFFLEVELS	= <*boolean*>		
ISVERTICAL	= <*boolean*>		
LABELINCREMENT	= <*double*>		
LABELLOCATION	= <*contourlabellocation*>		
NUMBERTEXTSHAPE	<<textshape>>		
OVERLAYBARGRID	= <*boolean*>		Thin line around each band in the color bar.
ROWSPACING	<*op*> <*dexp*>		
SHOW	= <*boolean*>		
SHOWHEADER	= <*boolean*>		
TEXTCOLOR	= <*color*>		
XYPOS	<<xy>>		
}			
VAR	= <*integer*>		Variable used for contour levels.

Example: This example does the following: Turns on the contour legend; Sets the flood cut-off 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
}
}
```

$!GLOBALEDGE

Syntax: `$!GLOBALEDGE`

Description: A SetValue command that sets attributes which sets the minimum crease angle for edges.

Optional Parameters:

Parameter	Syntax	Default
`MINCREASEANGLE`	*<double>*	135

$!GLOBALFRAME

Syntax: `$!GLOBALFRAME`
[optional parameters]

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

Optional Parameters:

Parameter	Syntax	Notes
`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)."
`FRAMEHEADERHEIGHT`	*<op>* *<dexp>*	Value is in inches.
`SNAPTOGRID`	= *<boolean>*	Even if set to TRUE, Tecplot may not allow snapping in some situations.
`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
```

$!GLOBALLINEPLOT

Syntax: `$!GLOBALLINEPLOT`

[optional parameters]

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

Optional Parameters:

Parameter	Syntax	Notes
DATALABELS **{** COLOR COLORBYZONEMAP DISTANCESKIP INCLUDEBOX INDEXSKIP NODELABELTYPE NUMFORMAT SHOWNODELABELS SKIPMODE TEXTSHAPE **}**	 = <*color*> = <*boolean*> <*op*> <*dexp*> = <*boolean*> <*op*> <*integer*> = <*labeltype*> <<numberformat>> = <*boolean*> = <*skipmode*> <<textshape>>	These are text values that can be added to a plot to show the indices or values for the data points. Not allowed to change size units parameter.
LEGEND **{** ANCHORALIGNMENT BOX ROWSPACING SHOW SHOWTEXT TEXTCOLOR TEXTSHAPE XYPOS **}**	 = <anchoralignment> <<textbox>> <*op*> <*dexp*> = <*boolean*> = <*boolean*> = <*color*> <<textshape>> <<xy>>	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. Not allowed to change size units.

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

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>*	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	Default	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>*	0.0	
RANGEMAX	= *<double>*	1.0	
LEGEND **{** AHCHOR BLUECHANNELLABEL BOX GREENCHANNELLABEL HEIGHT REDCHANNELLABEL RGBLEGENDORIENTATION SHOW SHOWLABELS TEXTCOLOR TEXTSHAPE USEBLUEVARNAME USEGREENVARNAME USEREDVARNAME XYPOS **}**	 = *<anchoralignment>* = *<string>* <<textbox>> = *<string>* = *<double>* = *<string>* = *<rgblegendorientation>* = *<boolean>* = *<boolean>* = *<color>* <<textshape>> = *<boolean>* = *<boolean>* = *<boolean>* <<xy>>		

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
BASEFONT	= <font>	
DATALABELS		These are text labels that can be added to a plot to show node or cell values.
{		
CELLLABELTYPE	= <labeltype>	
CELLLABELVAR	= <integer>	
COLOR	= <color>	
COLORBYZONEMAP	= <boolean>	
DISTANCESKIP	<op> <dexp>	
INCLUDEBOX	= <boolean>	
INDEXSKIP	<op> <integer>	
NODELABELTYPE	= <labeltype>	
NODELABELVAR	<op> <integer>	
NUMFORMAT	<<numberformat>>	
SHOWCELLLABELS	= <boolean>	
SHOWNODELABELS	= <boolean>	
SKIPMODE	= <skipmode>	
TEXTSHAPE	<<textshape>>	Not allowed to change size units parameter
}		
LEGEND		
{		
ANCHORPOS	<<anchorpos>>	
BOX	<<textbox>>	
ROWSPACING	<op> <dexp>	
SHOW	= <boolean>	
SHOWTEXT	= <boolean>	
TEXTCOLOR	= <color>	
TEXTSHAPE	<<textshape>>	Not allowed to change size units parameter.
XYPOS	<<xy>>	
}		
REFSCATSYMBOL		
{		
COLOR	= <color>	
FILLCOLOR	= <color>	
ISFILLED	= <boolean>	
LINETHICKNESS	<op> <dexp>	
MAGNITUDE	<op> <dexp>	
SHOW	= <boolean>	
SYMBOLSHAPE	<<symbolshape>>	
XYPOS	<<xy>>	
}		
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.
SPHERESCATTERRENDERQUALITY	= <spherescatterrenderquality>	Config file only option.
VAR	= <integer>	Scatter sizing variable.

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
}
```

$!GLOBALTHREED

Syntax: `$!GLOBALTHREED`
[optional parameters]

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

Optional Parameters:

Parameter	Syntax	Default	Notes
AXISBOXPADDING	<op> <dexp>		
AXISSCALEFACT	<<xyz>>		The 3-D axis must be INDEPENDENT for this option to work properly. See $!THREEDAXIS.

Parameter	Syntax	Default	Notes
FORCEGOURADFOR3DCONTFLOOD	= <*boolean*>	TRUE	
FORCEPANELEDFOR3DCELLFLOOD	= <*boolean*>	TRUE	
LIGHTSOURCE **{** BACKGROUNDLIGHT INCLUDESPECULAR INTENSITY SPECULARINTENSITY SPECULARSHININESS SURFACECOLORCONTRAST XYZDIRECTION **}**	 = <*double*> = <*boolean*> = <*double*> = <*integer*> = <*integer*> = <*double*> <<xyz>>	 FALSE	 Range = 1-100 Range = 1-100 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.
LINELIFTFRACTION	<*op*> <*dexp*>		
PERFORMEXTRA3DSORTING	<*boolean*>		
ROTATEORIGIN	<<xyz>>		
SLICE **{** ORIGIN NORMAL **}**	 <<xyz>> <<xyz>>		
SYMBOLLIFTFRACTION	<*op*> <*dexp*>		
VECTORLIFTFRACTION	<*op*> <*dexp*>		

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
ARROWHEADANGLE	<op> <dexp>	Angle is in degrees.
HEADSIZEASFRACTION	<op> <dexp>	Head is sized as a fraction of the stem length.
HEADSIZEINFRAMEUNITS	<op> <dexp>	Value is in Y-frame units.
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>	
RELATIVELENGTH	<op> <dexp>	
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.
SIZEHEADBYFRACTION	= <boolean>	If **TRUE**, **HEADSIZEASFRACTION** is used to size arrowheads otherwise **HEADSIZEINFRAMEUNITS** is used.
UNIFORMLENGTH	<op> <dexp>	Value is in Y-frame units.
USERELATIVE	= <boolean>	If **FALSE**, vectors are all the same size (**UNIFORMLENGTH**).
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.

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
}
```

$!GLOBALTIME

Syntax: **$!GLOBALTIME SolutionTime**

Description: A SetValue command for frames (2D and 3D ONLY). Different frames can have different values of **$!GLOBALTIME**

Parameter	Syntax	Default	Notes
CURSTEP	= *<integer>*		The value of the current time step
END	= *<double>*		
SKIP	= *<integer>*		
SOLUTIONTIME	= *<double>*	1	If SolutionTime is not set to a solution time in set of solution times from all zones of the active strand field-maps, SolutionTime is adjusted to the closest value in that set.
START	= *<double>*		

$!GLOBALTWODVECTOR

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

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

Optional Parameters:

Parameter	Syntax	Notes
ARROWHEADANGLE	*<op>* *<dexp>*	Angle is in degrees.
HEADSIZEASFRACTION	*<op>* *<dexp>*	Head is sized as a fraction of the stem length.
HEADSIZEINFRAMEUNITS	*<op>* *<dexp>*	Value is in Y-frame units.
REFVECTOR **{**		
SHOW	= *<boolean>*	
COLOR	= *<color>*	
MAGNITUDE	*<op>* *<dexp>*	
LINETHICKNESS	*<op>* *<dexp>*	

Parameter	Syntax	Notes
ANGLE XYPOS MAGNITUDELABEL { SHOW TEXTCOLOR TEXTSHAPE NUMFORMAT OFFSET } }	<op> <dexp> <<xy>> = <boolean> = <color> <<textshape>> <<numberformat>> = <double>	
RELATIVELENGTH	<op> <dexp>	
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.
SIZEHEADBYFRACTION	= <boolean>	If **TRUE**, **HEADSIZEASFRACTION** is used to size arrowheads otherwise **HEADSIZEINFRAMEUNITS** is used.
UNIFORMLENGTH	<op> <dexp>	Value is in Y-frame units.
USERELATIVE	= <boolean>	If **FALSE**, vectors are all the same size (**UNIFORMLENGTH**).
UVAR	= <integer>	Variable number for the X-vector component.
VVAR	= <integer>	Variable number for the Y-vector component.

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 VOLUMECELLINTERPOLATIONMODE CONTLINECREATEMODE **}**	 = <boundarycondition> <op> <integer> <op> <dexp> <op> <dexp> <op> <dexp> <op> <dexp> = <linearinterpmode> = <pointselection> <op> <integer> <op> <dexp> <op> <dexp> = <krigdrift> = <derivpos> <op> <dexp> = *[ACCURATE or FAST]* = *TRILINEAR* = *[ONEZONEPERCONTOURLEVER, ONEZONEPERINDEPENDENTPOLYLINE]*	

Parameter	Syntax	Notes
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.
{		
ADVANCED3DCONTROLDIALOG	<<dialogplacement>>	
ANCHORALIGNMENT	<<dialogplacement>>	
ANCHORHORIZONTALINSIDE	<<dialogplacement>>	
ANCHORVERTICALINSIDE	<<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>>	
IOFFSET	<<dialogplacement>>	
JOFFSET	<<dialogplacement>>	
ISOSURFACESDIALOG	<<dialogplacement>>	
KRIGINGINTERPOLATIONDIALOG	<<dialogplacement>>	
LIGHTSOURCEDIALOG	<<dialogplacement>>	
LINEARINTERPOLATIONDIALOG	<<dialogplacement>>	
LINEMAPLEGENDDIALOG	<<dialogplacement>>	
LOADDATADIALOG	<<dialogplacement>>	
MACROPLAYDIALOG	<<dialogplacement>>	
MACRORECORDDIALOG	<<dialogplacement>>	
MACROVIEWERDIALOG	<<dialogplacement>>	

Parameter	Syntax	Notes
MINVISIBILITYPERCENTAGE	<<dialogplacement>>	
MIRRORZONEDIALOG	<<dialogplacement>>	
NEWLAYOUTDIALOG	<<dialogplacement>>	
OPENLAYOUTDIALOG	<<dialogplacement>>	
ORDERFRAMESDIALOG	<<dialogplacement>>	
PAPERSETUPDIALOG	<<dialogplacement>>	
POLARDRAWINGOPTIONSDIALOG	<<dialogplacement>>	
POSITIONATANCHOR	<<dialogplacement>>	
PRINTDIALOG	<<dialogplacement>>	
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>>	
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.
INTIALFIELDPROBEDIALOGPAGE	= *[NODALVALUES, ZONECELLINFO, CELLCENTEREDVALUES, or FACENEIGHBORS]*	

Parameter	Syntax	Notes
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.
MAXNUMUNDOLEVELS	= *<integer>*	
MINPIXELSFORDRAG	*<integer>*	Number of pixels to move the pointer before it is considered a drag.
MOUSEACTIONS **{** MIDDLEBUTTON { BUTTONCLICK SIMPLEDRAG CONTROLLEDDRAG ALTEDDRAG SHIFTEDDRAG CONTROLALTEDDRAG CONTROLSHIFTEDDRAG ALTSHIFTEDDRAG CONTROLALTSHIFTEDDRAG } RIGHTBUTTON { BUTTONCLICK SIMPLEDRAG CONTROLLEDDRAG ALTEDDRAG SHIFTEDDRAG CONTROLALTEDDRAG CONTROLSHIFTEDDRAG	 *<mousebuttonclick>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>* *<mousebuttondrag>*	

Parameter	Syntax	Notes
ALTSHIFTEDDRAG CONTROLALTSHIFTEDDRAG } }	<*mousebuttondrag*> <*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.
OKTOEXECUTESYSTEMCOMMAND	= <*boolean*>	Allow use of $!SYSTEM commands in macros. This is a security issue. If set to FALSE and the macro is run intermittently 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 negligible.

Parameter	Syntax	Notes
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. Setting this above three is not recommended, 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.
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		Settings for interactive rotations in 3-D.
{		
ROTATIONMODE	= <*rotationmode*>	
CURRENTANGLE	= <*op*> <*dexp*>	
SMALLANGLE	= <*op*> <*dexp*>	
MEDIUMANGLE	= <*op*> <*dexp*>	
LARGEANGLE	= <*op*> <*dexp*>	
ROTATEDEGPERFRAMEUNIT	= <*integer*>	
SHOWGEOMS	= <*boolean*>	

Parameter	Syntax	Notes
}		
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.
SHOWSTATUSLINE	= *<boolean>*	
SHOWTEXTGEOMSINAPPROXVIEWS	= *<boolean>*	Set to TRUE if you want text and geometries to show up in frames using approximated plots
SHOWTOOLTIPS	= *<boolean>*	
SHOWWAITDIALOGS	= *<boolean>*	If FALSE, all "Please Wait" and "Percent Done" dialogs will be disabled.
SIDEBARSIZING	= *<sidebarsizing>*	
TRANSLATION **{** STEPSIZE SMALLSTEP MEDIUMSTEP LARGESTEP ZOOMSCALEPERFRAMEUNIT **}**	 *<op>* *<dexp>* *<op>* *<dexp>* *<op>* *<dexp>* *<op>* *<dexp>* = *<double>*	Settings for interactive translation.
TRUETYPEMINOUTLINEPOINTSIZE	= *<integer>*	

Parameter	Syntax	Notes
USEMOD2MASKFORALTDETECTION	= *<boolean>*	Certain platforms have a problem with the ALT key. Set to TRUE to bypass the problem.
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.
UNIXTRUETYPEFONTPATH	= *<string>*	Path to where true type fonts are stored.
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>*	
USEDISPLAYLISTS	= *<boolean>*	Use stroke fonts for data labels and ASCII scatter symbols in 3-D plots.
USEOFFSCREENBITMAP	= *<boolean>*	Set to TRUE to render images off-screen.
USESTROKEFONTSFOR3DTEXT	= *<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.
USESTROKEFONTSFORSMALLSCREENTEXT	= *<boolean>*	When using True Type fonts, switch to stroke fonts for small characters.
USESTROKEFONTSONSCREEN	= *<boolean>*	This applies to Windows only. Set to TRUE to use Tecplot's printer drivers. Set to FALSE to use Windows printer drivers.
USETECPLOTPRINTDRIVERS	= *<boolean>*	
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
INTERPNPOINTS	= <integer>	8	
INTERPPTSELECTION	= <interpptselection>	OCTANTNPOINTS	
INVDISTEXPONENT	= <dexp>	3.5	

Parameters	Syntax	Default	Notes
INVDISTMINRADIUS	= *<dexp>*	0.0	
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.

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

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

$!ISOSURFACEATTRIBUTES

Syntax: **$!ISOSURFACEATTRIBUTES** [<group]
[*optional parameters****]***

Description: A SetValue command which changes attributes associated with iso-surfaces. The optional group parameter can range from 1-4 and defaults to 1 when absent.

Optional Parameters:

Parameter	Syntax	Default	Notes
SHOWGROUP	= *<boolean>*		
ISOSURFACESELECTION	= *<isosurfaceselection>*		
ISOVALUE1	= *<double>*		
ISOVALUE2	= *<double>*		
ISOVALUE3	= *<double>*		
MESH **{** SHOW COLOR LINETHICKNESS **}**	 = *<boolean>* = *<color>* = *<double>*		
CONTOUR **{** SHOW USELIGHTINGEFFECT CONTOURTYPE FLOODCOLORING LINECONTOURGROUP COLOR LINETHICKNESS	 = *<boolean>* = *<boolean>* = *<contourtype>* = *<contourcoloring>* = *<integer>* = *<color>* = *<double>*	 FLOOD Group1	 PRIMARYVALUE and AVERAGECELL not allowed.

Parameter	Syntax	Default	Notes
}			
EFFECTS { LIGHTINGEFFECT SURFACETRANSLUCENCY USETRANSLUCENCY }	 = <*lightingeffect*> = <*translucency*> = <*boolean*>		
DEFINITIONCONTOURGROUP	= <*integer*>	1	Contour group from which iso-surfaces are based.
MARCHINGCUBEALGORITHM	= *[classic or classicplus]*		
OBEYSOURCEZONEBLANKING	= <*boolean*>		
SHADE { COLOR SHOW USELIGHTINGEFFECT }	 = <*color*> = <*boolean*> = <*boolean*>		

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}
```

$!ISOSURFACELAYERS

Syntax: **$!ISOSURFACELAYERS**

Required Parameters:

Parameter	Syntax	Notes
SHOW	<*boolean*>	

$!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
INTERPNPOINTS	= *<integer>*	8	
INTERPPTSELECTION	= *<interpptselection>*	OCTANTNPOINTS	
KRIGDRIFT	= *<krigdrift>*	LINEAR	
KRIGRANGE	= *<dexp>*	0.3	
KRIGZEROVALUE	= *<dexp>*	0.0	
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 3D) are not allowed.

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; 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
LODThresholdMinFract	<op> <double>	
LODThresholdMaxFract	<op> <double>	
MAXAVAILABLEPROCESSORS	<op> <integer>	Sets the maximum number of processors used by Tecplot. Some tasks can be performed in parallel so using all available processors can greatly increases performance of those tasks. A value of zero instructs Tecplot to use the maximum number of processors available on the machine up to the limit of 8 and provides the best performance in most cases. Values between 1 and 8 can be assigned to override what Tecplot thinks is the maximum number or to limit the number of processors used by Tecplot.
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.
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.
MAXTHREADS	<op> <integer>	Limit the number of threads
MAXUSABLEMEMORY	<op> <integer>	Limit the amount of memory used by Tecplot.

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	Default	Notes
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>*		
}			
BARCHARTS			
{			
SHOW	= *<boolean>*		

Parameter	Syntax	Default	Notes
COLOR	= <color>		
FILLMODE	= <fillmode>		
FILLCOLOR	= <color>		
SIZE	<op> <dexp>		
LINETHICKNESS	<op> <dexp>		
}			
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>		
}			
ERRORBARS			
{			
SHOW	= <boolean>		
VAR	= <integer>		
BARTYPE	= <errorbartype>		
COLOR	= <color>		
LINETHICKNESS	<op> <dexp>		
SKIPPING	<op> <dexp>		Skip can be by index or distance depending on SKIPMODE.
SKIPMODE	= <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>>		
}			
LINES			
{			
SHOW	= <boolean>		
COLOR	= <color>		
LINEPATTERN	= <linepattern>		
PATTERNLENGTH	= <op> <dexp>		

Parameter	Syntax	Default	Notes
LINETHICKNESS }	<op> <dexp>		
NAME	= <string>		
SYMBOLS { SHOW COLOR FILLMODE FILLCOLOR SIZE LINETHICKNESS SKIPPING SKIPMODE SYMBOLSHAPE }	 = <boolean> = <color> = <fillmode> = <color> <op> <dexp> <op> <dexp> <op> <dexp> = <skipmode> <<symbolshape>>		Skip can be by index or distance depending on SKIPMODE.

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
```

$!LINKCOLORMAPS

Syntax: `$!LINKCOLORMAPS` = <boolean>

Description: Set to true to tie all colormaps together.

$!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
BETWEENFRAMES { LINKCONTOURLEVELS LINKFRAMESIZEANDPOSITION	= <boolean>	

Parameter	Syntax	Notes
LINKXAXISRANGE	= <boolean>	
LINKYAXISRANGE	= <boolean>	
LINKPOLARVIEW	= <boolean>	
LINK3DVIEW	= <boolean>	
LINKGROUP	= <integer>	
LINKAXISPOSITION	= <boolean>	
LINKVALUEBLANKING	= <boolean>	
LINKSLICEPOSITIONS	= <boolean>	
LINKISOSURFACEVALUES	= <boolean>	
}		
WITHINFRAME		
{		
LINKAXISSTYLE	= <boolean>	
LINKGRIDLINESTYLE	= <boolean>	
LINKLAYERLINECOLOR	= <boolean>	
LINKLAYERLINEPATTERN	= <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
ADDONSTYLE	= <string>	V7Standard	Style of the add-on to load. This can be either V7STANDARD or V7ACTIVEX.
INITFUNCTION	= <string>	InitTecAddOn	Name of the function inside of the add-on that is used to initialize the add-on.

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
KEYSTROKE	= *<string>*		Allows keyboard shortcuts
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.

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>*	
GRIDSPACING	= *<papergridspacing>*	Set the spacing for the tick marks on the paper.
ISTRANSPARENT	= *<boolean>*	
ORIENTPORTRAIT	= *<boolean>*	
PAPERGRIDSPACING	= *<papergridspacing>*	
PAPERSIZE	= *<papersize>*	

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

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.
CONSIDERSTYLE	**=** *<boolean>*	FALSE	

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

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.
ARROWHEADATTACHMENT	**=** *<arrowheadattachment>*	
ARROWHEADSIZE	**=** *<dexp>*	Value is in Y-frame units (0-100).
ARROWHEADSTYLE	<arrowheadstyle>	

Parameters	Syntax	Notes
ASCIICHAR	= *<string>*	
BARCHARTS **{** SHOW ISFILLED **}**	 = *<boolean>* = *<boolean>*	Only operates on XY line mapping objects.
COLOR	= *<color>*	
CONTOUR { SHOW CONTOURTYPE **}**	 = *<boolean>* = *<contourtype>*	Only operates on 2- or 3-D zone objects.
CURVES **{** CURVETYPE **}**	 = *<curvetype>*	Only operates on XY line mapping objects.
EDGELAYER **{** SHOW SUBBOUNDARY **}**	 = *<boolean>* = *<subboundary>*	Only operates on 2- or 3-D zone objects.
ERRORBARS **{** SHOW BARTYPE **}**	 = *<boolean>* = *<errorbartype>*	Only operates on XY line mapping objects.
FILLCOLOR	= *<color>*	
FONT	= *<font>*	
GEOMSHAPE	= *<geomshape>*	Applies only to scatter symbols or XY-plot symbols.
LINEPATTERN	= *<linepattern>*	
LINES **{** SHOW **}**	 = *<boolean>*	Only operates on XY line mapping objects.
LINETHICKNESS	= *<dexp>*	Value is in Y-frame units (0-100).
MESH **{** SHOW MESHTYPE **}**	 = *<boolean>* = *<meshtype>*	Only operates on 2- or 3-D zone objects. Only operates on 2- or 3-D zone objects.
OBJECTALIGN	= *<objectalign>*	Only allowed if selected objects are all text and/or geometries.
PATTERNLENGTH	= *<dexp>*	Value is in Y-frame units (0-100).
SCATTER **{** SHOW FILLMODE **}**	 = *<boolean>* = *<fillmode>*	Only operates on 2- or 3-D zone objects.

Parameters	Syntax	Notes
SHADE **{** SHOW SHADETYPE **}**	 = <*boolean*> = <*shadetype*>	Only operates on 2- or 3-D zone objects.
SHOWBORDER	= <*boolean*>	Only operates on frame objects.
SIZE	= <*dexp*>	Value is in Y-frame units. This applies to things like symbols.
SYMBOLS **{** SHOW ISFILLED **}**	 = <*boolean*> = <*boolean*>	Only operates on line mapping objects.
TEXTCOLOR	= <*color*>	
TEXTHEIGHTBYPERCENT	= <*dexp*>	Value is in Y-frame units (0-100).
TEXTHEIGHTBYPOINTS	= <*dexp*>	Value is in points.
VECTOR **{** SHOW VECTORTYPE **}**	 = <*boolean*> = <*vectortype*>	Only operates on 2- or 3-D zone 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
GRIDAREA	<<areastyle>>	
PRECISEGRID	<<precisegrid>>	
PRESERVEAXISSCALE	<*boolean*>	
RDETAIL	<<axisdetail>>	
THETADETAIL	<<axisdetail>>	
THETAMODE	= <*thetamode*>	
THETAPERIOD	= <*double*>	
VIEWPORTPOSITION	<<rect>>	
VIEWPORTSTYLE	<<areastyle>>	

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	Default	Notes
DRIVER	= *<printerdriver>*		Only applies if using the Tecplot printer drivers. *See* **$!INTERFACE USETECPLOTPRINTDRIVERS**.
FORCEEXTRA3DSORTING	= *<boolean>*		
JOBCONTROL **{** POSTMOPUPSTR LGMOPUPSTR POSTSETUPSTR LGSETUPSTR **}**	 **=** *<string>* **=** *<string>* **=** *<string>* **=** *<string>*		These strings contain characters to be sent at the beginning and ending of a print file. These strings most often contain escape sequences used to switch modes on the printer. Non-printable characters can be inserted. Use ^*nnn* to insert a character with ordinal value *nnn*. Use \ to force the character after the \ to be inserted. Use **$B** for a Backspace, **$E** for Esc, **$C** for a carriage return, and **$X** for the Delete key.
IMAGERESOLUTION	= *<integer>*		
NUMHARDCOPYCOPIES	*<op>* *<integer>*		Applies only when **DRIVER** = **PS.**
NUMLIGHTSOURCESHADES	= *<integer>*		
PALETTE	**=** *<palette>*		Must choose options valid for current **DRIVER** setting.
PRECISION	*<op>* *<integer>*		Applies only if **EXPORTFORMAT** is **PS**, **EPS**, or **RASTERMETAFILE**.
PRINTFNAME	**=** *<string>*		Name of the file to write to if **SENDPRINTTOFILE** is **TRUE.**
PRINTRENDERTYPE	= *<printrendertype>*		
RGBLEGENDOUTPUTRESOLUTION	= *<integer>*	50	Determines the number of triangles which compose the bottom layer of the RGB Legend. This option is only available through macro language (for example, the config file)
SENDPRINTTOFILE	**=** *<boolean>*		If **TRUE** then **PRINTFNAME** is name of file to write to.

Parameter	Syntax	Default	Notes
SPOOLER **{** **PSMONOSPOOLCMD** **PSCOLORSPOOLCMD** **LGSPOOLCMD** **}**	 **=** *<string>* **=** *<string>* **=** *<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.
USEISOLATIN1FONTS-INPS	= *<boolean>*		Use extended ISO-Latin1 fonts when generating PostScript output using Tecplot's internal PostScript driver.

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.

Parameter	Syntax	Default	Notes
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
FRAMECOLLECTION	*= [ALL, PICKED]*	
LINKTYPE	*= [WITHINFRAME, BETWEENFRAMES]*	

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

Example:

```
$!PUBLISH "C:\TEC360\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
```

```
100100          100
125125          125
150150          150
175175          175
200200          200
225225          225
255255          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
ADDZONETOEXISTINGSTRANDS	= *<boolean>*	FALSE	If TRUE, Tecplot will add the zones from the appended data to any existing strands in the dataset. If FALSE, Tecplot will append the strands from the appended data to any existing strands in the dataset.
ASSIGNSTRANDID	= *<boolean>*	FALSE	If TRUE, Tecplot will assign strand ID's to zones if time is supplied for the zones but strand ID's are not. If FALSE, Tecplot will not associate these zones with any strands.
IJKSKIP **{** I J K **}**	 = *<integer>* = *<integer>* = *<integer>*	 1 1 1	Use values greater than 1 to skip data points.
COLLAPSEZONESANDVARS	= *<boolean>*	FALSE	Renumber zones and variables if zones or variables are disabled.
DATASETREADER	= *<string>*		Used to specify an alternate data reader for Tecplot.
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.
INCLUDEGEOM	= *<boolean>*	TRUE	Set to TRUE to load in any geometries in the data files.
INCLUDETEXT	= *<boolean>*	TRUE	Set to TRUE to load in any text in the data files.

Parameters	Syntax	Default	Notes
INITIALPLOTFIRSTZONEONLY	= *<boolean>*		Allows faster performance for files with multiple zones.
INITIALPLOTTYPE	= *<plottype>*		
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.
RESETSTYLE	= *<boolean>*	TRUE	Set to FALSE if you want Tecplot to keep the current style. This only applies if READDATA OPTION is not APPEND.
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>*		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.

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
`INCLUDEAUXDATA`	= *<boolean>*	TRUE	Set to `TRUE` to read auxiliary data.
`INCLUDECONTOURLEVELS`	= *<boolean>*	TRUE	Set to `TRUE` to read in all contour levels.
`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.
`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.
`INCLUDETEXT`	= *<boolean>*	TRUE	Set to `TRUE` to load in any text in the stylesheet file.
`MERGE`	= *<boolean>*	FALSE	Set to `FALSE` to reset all frame attributes back to their factory defaults prior to reading in the stylesheet.

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.**
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
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.
USERELATIVEPATHS	= *<boolean>*	FALSE	If TRUE, all files referenced in the layout file will use relative paths.

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 Auxiliary 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 by one or more underscore, period, letter, or digit characters.

Required Parameters:

Parameter	Syntax	Notes
AUXDATALOCATION	= *[zone/var/dataset/frame/linemap]*	
NAME	= *<string>*	
VALUESTRING	= *<string>*	

Optional Parameters:

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

Example: Set the selected Auxiliary 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
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.
FIELDVALUE	= <dexp>	
INDEX	= <integer>	
VAR	= <integer>	
ZONE	= <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
```

$!SETFRAMEBACKGROUNDCOLOR

Syntax: `$!SETFRAMEBACKGROUNDCOLOR` <color>

Description. Sets the frame background to the specified color and surveys all basic color assignments

in Tecplot, converting the all basic colors using the following rules to achieve the best contrast:

1. For all line type basic colors that match the new basic frame color, set the basic line color to the best show color of the basic frame color.
2. For all fill type basic colors that match the best show color of the new basic frame color, set the fill color to the new frame color.

Exceptions: 3.

1. For geometries and text boxes if the line and fill colors are the same and filling is active then both lines and fill follow the fill rules above.
2. For zone, slice, iso-surface, and streamtrace object types the basic color shading (i.e. fill) only follows the fill rules above if lighting effects are not being used.

$!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
DESTINATIONZONE	= <integer>	
SOURCEZONE	= <integer>	

Example: Shares the connectivity 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 source 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
DESTINATIONZONE	= <integer>	
SOURCEZONE	= <integer>	
VAR	= <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 deactivated 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	Default	Notes
AUTOADJUSTRANGESTONICEVALEUS	= *<boolean>*		
AXISMODE	<axismode>		Set to INDEPENDENT or XYDEPENDENT.
DEPXTOYRATIO	*<op>* *<dexp>*		AXISMODE must be XYDEPENDENT to use this.
GRIDAREASTYLE	*<<gridarea>>*		
PRECISEGRID	<<precisegrid>>		
PRESERVEAXISSCALEWHENRANGEISCHANGED	= *<boolean>*		
VIEWPORTNICEFITBUFFER	= *<double>*		
VIEWPORTPOSITION	<<rect>>		
VIEWPORTTOPSNAPTARGET	= *<integer>*	100	
VIEWPORTTOPSNAPTOLERANCE	= *<integer>*	10	
XDETAIL	<<axisdetail>>		
YDETAIL	<<axisdetail>>		

Example: Change the axis mode to be **INDEPENDENT** for sketch mode in the current frame:

```
$!SKETCHAXIS
AXISMODE = INDEPENDENT
```

$!SLICEATTRIBUTES

Syntax: **$!SLICEATTRIBUTES** [<slicegroup>]

[optional parameters]

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

Optional Parameters:

Parameter	Syntax	Default	Notes
CONTOUR			
{			
SHOW	= *<boolean>*		
CONTOURTYPE	= *<contourtype>*		CORNERCELL and AVERAGECELL options not allowed for CONTOURTYPE.
COLOR	= *<color>*		
LINETHICKNESS	= *<double>*		
USELIGHTINGEFFECT	= *<boolean>*		
FLOODCOLORING	= *<contourcoloring>*		
LINECONTOURGROUP	= *<integer>*	Group1	

Parameter	Syntax	Default	Notes
}			
EDGELAYER { SHOW COLOR LINETHICKNESS }	 = <boolean> = <color> = <op><double>		
ENDPOSITION { X Y Z I J K }	 = <double> = <double> = <double> = <integer> = <integer> = <integer>		
MESH { SHOW COLOR LINETHICKNESS }	 = <boolean> = <color> = <double>		
NUMINTERMEDIATESLICES	= <integer>		
OBEYSOURCEBLANKING	= <boolean>		
PRIMARYPOSITION { X Y Z I J K }	 = <double> = <double> = <double> = <integer> = <integer> = <integer>		
SHADE { SHOW COLOR USELIGHTINGEFFECT }	 = <boolean> = <color> = <boolean>		
SHOWGROUP	= <boolean>		
SHOWINTERMEDIATESLICES	= <boolean>		
SHOWPRIMARYSLICE	= <boolean>		
SHOWSTARTENDSLICE	= <boolean>		
SLICESURFACE	= <slicesurface>		
STARTPOSITION { X Y	 = <double> = <double>		

Parameter	Syntax	Default	Notes
Z	= <double>		
I	= <integer>		
J	= <integer>		
K	= <integer>		
}			
SURFACEEFFECTS			
{			
LIGHTINGEFFECT	=<lightingeffect>		
SURFACETRANSLUCENCY	= <translucency>		
USETRANSLUCENCY	= <boolean>		
}			
VECTOR			
{			
SHOW	= <boolean>		
COLOR	= <color>		
ISTANGENT	= <boolean>		
LINETHICKNESS	= <double>		
VECTORTYPE	= <vectortype>		
ARROWHEADSTYLE	<arrowheadstyle>		
}			

Example:

```
$!GLOBALCONTOUR VAR = 4
$!SLICEATTRIBUTES ENDPOSITION {X = 1}
$!SLICEATTRIBUTES STARTPOSITION {X = 6}
$!SLICEATTRIBUTES NUMITERMEDIATESLICES = 6
$!SLICEATTRIBUTES SHOWBEGINENDSLICE = YES
$!SLICEATTRIBUTES SHOWINTERMEDIATESLICES = YES
$!REDRAW
$!CREATESLICEZONES
```

$!SLICELAYERS

Syntax: **$!SLICELAYERS**

Required Parameters:

Parameter	Syntax	Notes
SHOW	<boolean>	

$!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
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]
```

$!STREAMATTRIBUTES

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

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

Optional Parameters:

Parameter	Syntax	Notes
ADDARROWS	= <*boolean*>	
ARROWHEADSIZE	<*op*> <*dexp*>	
ARROWHEADSPACING	<*op*> <*double*>	Distance between arrowheads in frame units.
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.
COLOR	= <*color*>	
LINETHICKNESS	<*op*> <*dexp*>	
MAXSTEPS	<*op*> <*integer*>	
MINCELLFRACTION	<*op*> <*dexp*>	Minimum fraction of the distance across a cell a streamtrace moves in one step.

Parameter	Syntax	Notes
OBEYSOURCEZONEBLANKING	= <*boolean*>	
RODRIBBON **{** WIDTH NUMRODPOINTS MESH { SHOW COLOR LINETHICKNESS } CONTOUR { SHOW USELIGHTINGEFFECT FLOODCOLORING } SHADE { SHOW COLOR USELIGHTINGEFFECT } SURFACEEFFECT { LIGHTINGEFFECT SURFACETRANSLUCENCY USETRANSLUCENCY } **}**	 <*op*><*dexp*> <*op*> <*integer*> = <*boolean*> = <*color*> <*op*><*dexp*> = <*boolean*> = <*boolean*> = <*contourcoloring*> = <*boolean*> = <*color*> = <*boolean*> = <*lightingeffect*> = <*translucency*> = <*boolean*> 	 Value is grid units. Number of points used to define the streamrod cross-section.
SHOW	**=** <*boolean*>	
STREAMTIMING **{** SHOWDASHES SHOWMARKERS MARKCOLOR MARKSIZE DASHSKIP MARKSYMBOL TIMESTART TIMEEND TIMEANCHOR TIMEDELTA **}**	 **=** <*boolean*> **=** <*boolean*> **=** <*color*> <*op*> <*dexp*> <*op*> <*integer*> <<symbolshape>> = <*double*> = <*double*> = <*double*> = <*double*>	
TERMLINE **{** ISACTIVE	 **=** <*boolean*>	

Parameter	Syntax	Notes
SHOW	= <boolean>	
COLOR	= <color>	
LINEPATTERN	= <linepattern>	
PATTERNLENGTH	<op> <dexp>	
LINETHICKNESS	<op> <dexp>	
}		

$!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
ALTSTARTPOS			This is required if NUMPTS is greater than 1 or if the streamtype is a volume rod or volume ribbon.
{			
X	= <dexp>	0.0	
Y	= <dexp>	0.0	
Z	= <dexp>	0.0	
}			
DIRECTION	= <streamdirection>	FORWARD	
NUMPTS	= <integer>	1	Use 1 to add a single streamtrace. Use *n*, *n*>1 for a rake of streamtraces.

Parameters	Syntax	Default	Notes
STARTPOS			Z is necessary only if dealing with a 3-D streamtrace.
{			
X	= *<dexp>*	0.0	
Y	= *<dexp>*	0.0	
Z	= *<dexp>*	0.0	
}			
STREAMTYPE	= *<streamtype>*	a	

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
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	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
4.07.0
5.09.0
5.03.0
```

$!STREAMTRACELAYERS

Syntax: **$!STREAMTRACELAYERS**

Required Parameters:

Parameter	Syntax	Notes
SHOW	*<boolean>*	

$!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
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.
AXISMODE	<*axismode*>	Set to INDEPENDENT, XYDEPENDENT, or XYZDEPENDENT.
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.
DEPXTOYRATIO	<*op*> <*dexp*>	AXISMODE must be DEPENDENT to use this.
DEPXTOZRATIO	<*op*> <*dexp*>	AXISMODE must be DEPENDENT to use this.
EDGEAUTORESET	= <*boolean*>	Make Tecplot automatically choose edges to label.
FRAMEAXIS **{** SHOW SIZE LINETHICKNESS COLOR XYPOS **}**	 = <*boolean*> <*op*> <*dexp*> <*op*> <*dexp*> = <*color*> <<*xy*>>	
GRIDAREA	<<*gridarea*>>	
PRESERVEAXISSCALEWHENRANGEISCHANGED	= <*boolean*>	
XDETAIL	<<*axisdetail*>>	
XYDEPXTOYRATIO	<*op*> <*dexp*>	AXISMODE must be XYDEPENDENT to use this.
YDETAIL	<<*axisdetail*>>	
ZDETAIL	<<*axisdetail*>>	

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
ALPHAANGLE	<op> <dexp>	Angle is in degrees.
DRAWINPERSPECTIVE	= <boolean>	
FIELDOFVIEW	<op> <dexp>	
PSIANGLE	<op> <dexp>	Angle is in degrees.
THETAANGLE	<op> <dexp>	Angle is in degrees.
VIEWERPOSITION	<<xyz>>	
VIEWWIDTH	<op> <dexp>	

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
 }
```

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
ANGLESPEC	<anglespec>	RADIANS	Specifies whether data is in degrees or radians
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.
PSIVAR	= *<integer>*		PSI variable number. REQUIRED if the transformation is spherical to rectangular or if **CREATENEWVARIABLES** is **FALSE**.
RVAR	= *<integer>*		R variable number. REQUIRED if the transformation is polar to rectangular or spherical to rectangular or if **CREATENEWVARIABLES** is **FALSE**.
THETAVAR	= *<integer>*	NONE	Theta variable number. REQUIRED if the transformation is polar to rectangular or 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**.
ZONELIST	= *<set>*	all zones	Set if zones to operate on.
ZVAR	= *<integer>*		Z variable number. REQUIRED if the transformation or rectangular to spherical or **CREATENEWVARIABLES** is **FALSE**.

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
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.
SOURCEZONES	= *<set>*	All zones.	
TRIANGLEKEEPFACTOR	= *<dexp>*	0.25	
USEBOUNDARY	= *<boolean>*	FALSE	Specify one or more I-ordered zones that define boundaries across which no triangles can be created.

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	Default	Notes
AUTOADJUSTRANGESTONICEVALUES	= *<boolean>*		
AXISMODE	*<axismode>*		Set to **INDEPENDENT** or **XYDEPENDENT**.
DEPXTOYRATIO	*<op>* *<dexp>*		**AXISMODE** must be **XYDEPENDENT** to use this.
GRIDAREA	*<<gridarea>>*		
PRECISEGRID	<<precisegrid>>		

Parameter	Syntax	Default	Notes
PRESERVEAXISSCALEWHENRANGEISCHANGED	= <boolean>		
VIEWPORTNICEFITBUFFER	= <double>		
VIEWPORTPOSTITION	<<rect>>		
VIEWPORTTOPSNAPTARGET	= <integer>	100	
VIEWPORTTOPSNAPTOLERANCE	= <integer>	10	
XDETAIL	<<axisdetail>>		
YDETAIL	<<axisdetail>>		

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 initial 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
ASSOCIATELAYOUTWITHDATAFILE	= *<boolean>*	TRUE	
BINARY	= *<boolean>*	TRUE	If FALSE, you can include PRECISION and USEPOINTFORMAT.
INCLUDEAUTOGENFACENEIGHBORS	= *<boolean>*	FALSE	
INCLUDECUSTOMLABELS	= *<boolean>*	TRUE	
INCLUDEDATA	= *<boolean>*	TRUE	
INCLUDEDATASHARELINKAGE	= *<boolean>*	FALSE	
INCLUDEGEOM	= *<boolean>*	TRUE	
INCLUDETEXT	= *<boolean>*	TRUE	
PRECISION	= *<integer>*	12	Only used if ASCII (that is, BINARY is FALSE).
USEPOINTFORMAT	= *<boolean>*	FALSE	Only used if ASCII (that is, BINARY is FALSE).
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.

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
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	Default	Notes
AUTOADJUSTRANGESTONICEVALUES	= *<boolean>*		
AXISMODE	*<axismode>*		Set to **INDEPENDENT** or **XYDEPENDENT**.
DEPXTOYRATIO	*<op>* *<dexp>*		**AXISMODE** must be **XYDEPENDENT** to use this. This applies only to the X1- and Y1-axes.
GRIDAREA	*<<gridarea>>*		
PRECISEGRID	<<precisegrid>>		
PRESERVEAXISSCALE	= *<boolean>*		
VIEWPORTNICEFITBUFFER	= *<double>*		Between 1 and 100.
VIEWPORTTOPSNAPTARGET	= *<integer>*	100	
VIEWPORTTOPSNAPTOLERANCE	= *<integer>*	10	
XDETAIL	*<integer>* <<axisdetail>>		The *<integer>* option specifies which axis to operate on, 1 £ n £ 5.
YDETAIL	*<integer>* <<axisdetail>>		The *<integer>* option specifies which axis to operate on, 1 £ *n* £ 5.

Example: Set the axis mode to be independent for the XY-axes (note that this affects only X1 versus Y1):

```
$!XYLINEAXIS
AXISMODE = INDEPENDENT
```


Chapter 9

Macro Commands for the Analyze Menu

All of macro commands associated with the **Analyze** menu are embedded within Tecplot's **ADDONCOMMAND** macro. The syntax of this macro is shown below:

```
$!ADDONCOMMAND
ADDONID = <string>
COMMAND = <string>
```

The first *<string>* is a text string should be set to **CFDAnalyzer3**. The second string is sent to one of the add-ons listed below.

9 - 1 Summary of Analyze Macro Commands

ANIMATESTREAKLINES may be used following a streakline calculation to animate the streaklines, either to the screen or to a file.

ATTACHINTEGRATIONRESULTS is used following an integration to create a text field and attach it to the current Tecplot frame. This macro has the same effect as clicking Make Text on the Integration Results text dialog.

Note: It is not necessary to direct the macro to display the Integration Results dialog in order to attach or save the results.

CALCPARTICLEPATH calculates particle paths or streaklines for steady or unsteady flow solutions, using the location of any existing streamtraces as starting locations for the particles. Particles may have mass or be massless.

CALCTURBULENCEFUNCTION calculates any of four turbulence-related functions, given any two in your data set.

CALCULATE calculates a PLOT3D function. The name of this function must be specified in the shortened form listed in Section 12.5, "Parameter Assignment Values."

CALCULATEACCURACY uses Richardson extrapolation to estimate the order accuracy of the solution, given the solution on three grids of successively finer resolution. If either of the plotting options are set to **TRUE**, the resulting Tecplot frames will be in front after executing this command.

DISPLAYBOUNDARIES displays zone boundaries in a new frame according to settings made by the **SETGEOMETRYANDBOUNDARIES** macro. Each boundary of each 3-D zone (in 3D Cartesian plots) or 2-D zone (in 2D Cartesian plots) is displayed and named according to the boundary condition applied to it. Boundaries that are connected to the boundaries of adjacent zones are named as such.

EXTRACTFLOWFEATURE displays shock surfaces, vortex cores, or separation and attachment lines for 3-D flow solutions. Separation and attachment lines are only calculated on no-slip wall boundaries identified by the **SETGEOMETRYANDBOUNDARIES** macro. Shock surfaces are displayed as iso-surfaces of a new variable, ShockFeature, while the remaining features are displayed as new zones.

EXTRAPOLATESOLUTION performs Richardson extrapolation to estimate the true solution from three input solutions on grids of successively finer resolution. It saves the extrapolated solution as a new zone in the current data set. It also saves an additional zone containing the difference between this solution and the original solution.

INTEGRATE performs an integration. All Integrate dialog options are available to this macro, including the display options. If the **PLOTRESULTS** parameter is set to **TRUE**, then the Tecplot frame showing the integration results is the current frame following this command.

SAVEINTEGRATIONRESULTS has the same effect as clicking Save on the Integration Results dialog and selecting a file. The results are saved to the file named by the **FILE** parameter.

SETFIELDVARIABLES identifies variables in your data, such as velocity, pressure and temperature, for use in analysis.

SETFLUIDPROPERTIES sets the properties of the fluid, such as viscosity. These are used by some actions of the **CALCULATE** and **INTEGRATE** commands.

SETGEOMETRYANDBOUNDARIES identifies boundaries of zones in a flow solution and the boundary conditions applied to them. It also specifies whether zones with coincident boundary nodes should be considered connected at those points, as well as whether 2-D solutions should be regarded as axisymmetric.

SETREFERENCEVALUES sets the reference (free-stream) properties of the solution. This information is used by other calculations.

SETUNSTEADYFLOWOPTIONS identifies solution time levels for unsteady flow solutions. This information is used for particle path and streakline calculations.

9 - 2 Macro Command Description

The syntax, mandatory and optional parameters for each of the macro commands listed in 9 - 1are described below. Items within single angle brackets (**<>**) are defined in Section 9 - 3.

Note: The **COMMAND** strings below must be contained on a single line in your macro command file, although they appear on multiple lines below.

ANIMATESTREAKLINES

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDANALYZER3'
COMMAND = 'ANIMATESTREAKLINES
    [optional parameters]'
```

Description: Animates previously calculated streaklines to the screen or to a file.

Optional Parameters:

Parameter	Syntax	Default	Notes
DESTINATION	= *<string>*	**SCREEN**	Specifies the destination of the animation. May be **SCREEN**, **AVIFILE** or **RASTERMETAFILE**.
FILENAME	= *<string>*	**""**	The name of the file to which to save the animation. Must be specified for **DESTINATION** values of **AVIFILE** or **RASTERMETAFILE**.
WIDTH	= *<integer>*	**300**	The width of the animation when saved to a file.
SPEED	= *<double>*	**10.0**	The speed in frames per second of the animation. Only used for animations saved to an AVI file.
USEMULTIPLECOLORTABLES	= *<boolean>*	**FALSE**	Specifies whether animations saved to a file should include one color table for each frame. The default is to use a single color table.
INCLUDEZONEANIMATION	= *<boolean>*	FALSE	

ATTACHINTEGRATIONRESULTS

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'ATTACHINTEGRATIONRESULTS'
```

Description: Attach the text results of the previous integration as a text field in the current frame.

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'CALCPARTICLEPATH
    [optional parameters]'
```

Description: Calculate particle paths or streaklines, starting from existing Tecplot streamtraces.

Optional Parameters:

Parameter	Syntax	Default	Notes
FUNCTION	= *<particlefunction>*	**PARTICLEPATH**	Can be **PARTICLEPATH** or **STREAKLINE**.
TIMESTEP	= *<double>*	**1**	The integration time step for the calculation.
MAXTIMESTEPS	= *<integer>*	**1000**	For steady-state calculations only.
RELEASEFREQ	= *<double>*	**1**	For **FUNCTION = STREAKLINE**. Indicates the number of particles to release in the indicated time period (see the next parameter).
RELEASEOPTION	= *<releaseoption>*	**TIMELEVEL**	For **FUNCTION = STREAKLINE**. If **TIMELEVEL**, indicates that **RELEASEFREQ** particles should be released every solution time level. If **UNITTIME**, indicates that this number of particles should be released in a unit amount of solution time.
HAVEMASS	= *<boolean>*	FALSE	If **TRUE**, particles have mass; specify the particle mass options below.
CREATESINGLEZONE	= *<boolean>*	FALSE	For **FUNCTION** = **PARTICLEPATH** only, specifies that all particle paths should be combined into a single I-J ordered zone.
STOREOPTION	= *<storeoption>*	PARTICLEVALUES	If **PARTICLEVALUES**, the particle's velocity, mass and temperature (if calculated) will be stored in place of appropriate fluid values in the particle path's zone. If **FLUIDVALUES**, all fluid values the particle passed through will be stored in the zone.

Parameter	Syntax	Default	Notes
COEFFS	= *<coeffsoption>*	GENERAL	If **GENERAL**, specify **BALLISTICCOEFF**, plus **TEMPTIMECONST** if calculating particle temperature. If **DETAILED**, specify **MASS**, **RADIUS**, and **DRAGCOEFF**, plus **SPECIFICHEAT** and **NUSSELT** if calculating temperature. Only applies if **HAVEMASS = TRUE**.
CALCTEMERATURE	= *<boolean>*	FALSE	If **TRUE**, particle temperature will be calculated. Only applies if **HAVEMASS = TRUE**.
GRAVITYCONSTANT	= *<double>*	0.0	The acceleration due to gravity. Only applies if **HAVEMASS = TRUE**.
GRAVITYDIRECTION	= *<gravitydirection>*	MINUSX	The axis direction in which gravity acts. Only applies if **HAVEMASS = TRUE**.
INITIALVELOCITYOPTION	= <*initialvelocityoption*>	LOCALFLUIDVELOCITY	The initial velocity of particles. Options are **LOCALFLUIDVELOCITY** and **ZEROVELOCITY**. Only applies if **HAVEMASS = TRUE**.
BALLISTICCOEFF	= *<double>*	1.0	For **GENERAL** coefficients only, the ballistic coefficient of the particle. Only applies if **HAVEMASS = TRUE.**
TEMPTIMECONST	= *<double>*	1.0	For **GENERAL** coefficients with **CALCTEMPERATURE** = **TRUE** only, the temperature relaxation factor of the particle. Only applies if **HAVEMASS = TRUE**.
MASS	= *<double>*	1.0	For **DETAILED** coefficients only, the particle mass. Only applies if **HAVEMASS = TRUE**.
RADIUS	= *<double>*	1.0	For **DETAILED** coefficients only, the particle initial radius. Only applies if **HAVEMASS = TRUE**.
DRAGCOEFFOPTION	= *<specifyoption>*	SPECIFY	For **DETAILED** coefficients only. If **SPECIFY**, specify **DRAGCOEFF**. If **CALCULATE**, Tecplot will calculate the drag coefficient. Only applies if **HAVEMASS = TRUE**.

Parameter	Syntax	Default	Notes
DRAGCOEFF	= *<double>*	1.0	For **DETAILED** coefficients only, with **DRAGCOEFFOPTION** = **SPECIFY**, the particle drag coefficient. Only applies if **HAVEMASS = TRUE.**
SPECIFICHEAT	= *<double>*	1.0	For **DETAILED** coefficients with **CALCTEMPERATURE** = **TRUE** only, the particle specific heat. Only applies if **HAVEMASS = TRUE.**
NUSSELTOPTION	= *<specifyoption>*	SPECIFY	For **DETAILED** coefficients with **CALCTEMPERATURE** = **TRUE** only. If **SPECIFY**, specify **NUSSELT**. If **CALCULATE**, Tecplot will calculate the Nusselt number. Only applies if **HAVEMASS = TRUE**.
NUSSELT	= *<double>*	1.0	For **DETAILED** coefficients with **CALCTEMPERATURE** = **TRUE** and **NUSSELTOPTION** = **SPECIFY** only, the particle Nusselt number. Only applies if **HAVEMASS = TRUE**.
TERMOPTION	= *<terminationoption>*	TEMPERATURE	For **DETAILED** coefficients with **CALCTEMPERATURE** = **TRUE** only (is always **TEMPERATURE** for general coefficients), the particle termination option. May be **TEMPERATURE** or **ABLATE**. Only applies if **HAVEMASS = TRUE**.
TEMPERATURE	= *<double>*	1.0	If **TERMOPTION** = **TEMPERATURE**, the particle termination temperature. If **TERMOPTION** = **ABLATE**, the ablation temperature. Only applies if **HAVEMASS = TRUE**.
LATENTHEAT	= *<double>*	1.0	For **TERMOPTION** = **ABLATE** only, the latent heat of the ablative process. Only applies if **HAVEMASS = TRUE**.

Example 1: Calculate streaklines with an integration time step of 0.1, releasing eight particles per unit solution time:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
```


```
COMMAND = 'CALCPARTICLEPATH
   FUNCTION=STREAKLINE
   TIMESTEP=0.1    RELEASEFREQ=8
   RELEASEOPTION=UNITTIME'
```

Example 2: Calculate particle paths, including temperature with ablation, in a steady-state flow for particles with an initial mass of 3E-14, an initial radius of 1.5E-6 and a specific heat of 703. Use a time step of 1E-6. Have Tecplot calculate the drag coefficient and the Nusselt number. Use an ablation temperature of 2,250 and a latent heat of 1.5E5:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'CALCPARTICLEPATH
   TIMESTEP = 1.0e-6
   HAVEMASS = TRUE
   COEFFS = DETAILED
   CALCTEMPERATURE = TRUE
   MASS = 3e-14
   RADIUS = 1.5e-6
   DRAGCOEFFOPTION = CALCULATE
   SPECIFICHEAT = 703
   NUSSELTOPTION = CALCULATE
   TERMOPTION = ABLATE
   TEMPERATURE = 2250
   LATENTHEAT = 1.5e5'
```

CALCTURBULENCEFUNCTION

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'CALCTURBULENCEFUNCTION
   [optional parameters]
```

Description: Calculate a turbulence-related function from two variables in the current data set. Add the result to the data set as a new variable using the function's name, or overwrite the variable if it already exists.

Optional Parameters:

Parameter	Syntax	Default	Notes
CALCULATEONDEMAND	= *<boolean>*	FALSE	
FUNCTION	= *<turbulencefunction>*	**FREQUENCY**	May be **ENERGY**, **DISSIPATIONRATE**, **FREQUENCY**, or **VISCOSITY**.
ID1	= *<turbulencefunction>*	**ENERGY**	The turbulence quantity the first data set variable represents.
VARIABLE1	= *<integer>*	**1**	The number of the first data set variable.
ID2	= *<turbulencefunction>*	**DISSIPATIONRATE**	The turbulence quantity the second data set variable represents.
VARIABLE2	= *<integer>*	**2**	The number of the second data set variable.
VALUELOCATION	= <*valuelocation*>	NODAL	The location of new variables added to the data set. Can be **NODAL** or **CELLCENTERED**.

Example: Calculate turbulent kinematic viscosity from turbulent kinetic energy, variable 5, and turbulent frequency, variable 6:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'CALCTURBULENCEFUNCTION
      FUNCTION=VISCOSITY
      VARIABLE1=5
      ID2=FREQUENCY VARIABLE2=6'
```

CALCULATE

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'CALCULATE FUNCTION = <functionname>
    [optional parameters]'
```

Description: Calculate a Tecplot variable using the specified function and add it to the current data set. If the variable already exists in the current data set, it will be recalculated.

Required Parameter:

Parameter	Syntax	Default	Notes
CALCULATEONDEMAND	= <*boolean*>	FALSE	
FUNCTION	= <*functionname*>		Indicates the function to be used to calculate the variable. If it is a vector function, the components will be stored as X *name*, Y *name*, and Z *name*, where *name* is the function name appearing in the interface.

Optional Parameter:

Parameter	Syntax	Default	Notes
NORMALIZATION	= <*normalizationoption*>	**NONE**	May be **NONE**, **MAXIMUMMAGNITUDE** or **REFERENCEVALUES**.
VALUELOCATION	= <*valuelocation*>	NODAL	The location of new variables added to the data set. Can be **NODAL** or **CELLCENTERED**.
CALCULATEONDEMAND	= <*boolean*>	FALSE	

Example 1: Calculate the Jacobian for the grid of the current data set:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'CALCULATE FUNCTION = JACOBIAN'
```

Example 2: Calculate the pressure coefficient for the current data set. The freestream density and speed of sound are 1.0 (the defaults):

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'CALCULATE
   FUNCTION = PRESSURECOEF
```

CALCULATEACCURACY

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'CALCULATEACCURACY ZONES = [<set>]
    [optional parameters]'
```

Description: Calculate the order accuracy of the solution contained in the listed zones. Optionally, plot the overall accuracy versus grid spacing and plot the accuracy at each grid node.

Required Parameter:

Parameter	Syntax	Notes
ZONES	= <set>	Indicates the three zones from which to perform the accuracy calculation.

Optional Parameters:

Parameter Syntax	Syntax	Default	Notes
MAXACCURACY	= <double>	2.0	The maximum theoretical accuracy of the solver which generated the solution. Used to limit the calculated accuracy.
DATASETVAR	= <integer>	1	The data set variable with which to perform the accuracy calculation.
PLOTDETAILEDACCURACY	= <boolean>	FALSE	If **TRUE**, a new frame will be created containing the accuracy calculated at each grid node.
PLOTOVERALLACCURACY	= <boolean>	FALSE	If **TRUE**, a new frame will be created containing the 1-norm and max-norm of the estimated error for each solution zone plotted versus grid resolution.

Example: Calculate the accuracy using zones 3, 4 and 5, along with data set variable 7, plotting the overall accuracy:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'CALCULATEACCURACY     ZONES=[3-5]
   DATASETVAR=7
   PLOTOVERALLACCURACY=TRUE'
```

DISPLAYBOUNDARIES

Syntax

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'DISPLAYBOUNDARIES
[optional parameters]
[RAWDATA
<boundaryrawdata>]
```

Description: Displays boundaries corresponding to a geometry and boundaries specification without actually setting the geometry and boundaries. This macro is generally not useful for those writing macro files, but is recorded when the user clicks the Display Boundaries button in the Geometry and Boundaries dialog in order to duplicate the actions of Tecplot that happen in response to that action. See "SETGEOMETRYANDBOUNDARIES" on page 241 for a description of the parameters for this macro.

EXTRACTFLOWFEATURE

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'EXTRACTFLOWFEATURE
    [optional parameters]'
```

Description: Extract and display shock surfaces, vortex cores, or separation and attachment lines. Shock surfaces are displayed as isosurfaces of a new variable, ShockSurface, while vortex cores and separation and attachment lines are displayed as new zones.

Optional Parameters:

Parameter	Syntax	Default	Notes
`Feature`	= <*flowfeature*>	**SHOCKSURFACES**	Can be **SHOCKSURFACES**, **VORTEXCORES**, or **SEPATTACHLINES**.
VCOREMETHOD	= <*vcoremethod*>	**EIGENMODES**	The vortex core extraction method. Can be **VORTICITY** or **EIGENMODES**.
EXCLUDEBLANKED	= <*boolean*>	FALSE	If **TRUE**, vortex cores and separation/ attachment lines will not be calculated in blanked regions.

Example: Extract vortex cores using the eigenmodes method:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'EXTRACTFLOWFEATURE
   FEATURE = VORTEXCORES
   VCOREMETHOD = EIGENMODES'
```

EXTRAPOLATESOLUTION

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'EXTRAPOLATESOLUTION ZONES = <set>
    [MAXACCURACY = <double>]'
```

Description: Perform Richardson extrapolation to estimate the true solution from three input solutions on grids of successively finer resolution. Two new zones are added to the current data set. The first contains the extrapolated solution, while the second contains the estimated error.

Required Parameter:

Parameter	Syntax	Notes
ZONES	= <*set*>	Indicates the three zones from which to perform the accuracy calculation.

Optional Parameter:

Parameter	Syntax	Default	Notes
MAXACCURACY	= <*double*>	**2.0**	The maximum theoretical accuracy of the solver which generated the solution. Used to limit the calculated accuracy.

Example: Extrapolate zones 3, 4, and 5, which were calculated with a second order accurate solver:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'EXTRAPOLATESOLUTION     ZONES=[3-5]
   MAXACCURACY = 2
```

INTEGRATE

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'INTEGRATE [<set>] [optional parameters]'
```

Description: Perform an integration over the specified zones. If ***<set>*** is not specified, the integration will be performed over all zones. If **PLOTAS** is set to **TRUE**, the integration results will be plotted in a new frame.

Optional Parameters:

Parameter	Syntax	Default	Notes
VARIABLEOPTION	= <*variableoption*>	**SCALAR**	
XORIGIN	= <*double*>	**0.0**	For **VARIABLEOPTION = FORCESANDMOMENTS**, indicates the origin X-location for moment calculations.
YORIGIN	= <*double*>	**0.0**	For **VARIABLEOPTION = FORCESANDMOMENTS**, indicates the origin Y-location for moment calculations.
ZORIGIN	= <*double*>	**0.0**	For **VARIABLEOPTION = FORCESANDMOMENTS**, indicates the origin Z-location for moment calculations.
SCALARVAR	= <*integer*>	**1**	For when VARIABLEOPTION = **AVERAGE,** MASSWEIGHTEDAVERAGE, WEIGHTEDAVERAGE, MASSFLOWWEIGHTEDAVERAGE, OR VECTORAVERAGe. Indicates which variable's average will be calculated.
ABSOLUTE	= <*boolean*>	**FALSE**	If **TRUE**, the absolute value of cell volumes be used for integration.

Parameter	Syntax	Default	Notes
EXCLUDEBLANKED	= *<boolean>*	**FALSE**	If **TRUE**, integration will only include non-blanked regions.
XVARIABLE	= *<integer>*	**0**	Data set position of the scalar variable or X-component of the vector variable to be integrated.
YVARIABLE	= *<integer>*	**0**	Only required for vector integrations. Indicates the Y-component of the vector variable to be integrated.
ZVARIABLE	= *<integer>*	**0**	Only required for vector integrations. Indicates the Z-component of the vector variable to be integrated.
INTEGRATEOVER	= *<integrationoption>*	**CELLVOLUMES**	Specifies cell volumes, planes, or lines.
IRANGE			
{			
MIN	= *<integer>*	1	
MAX	= *<integer>*	**0**	
SKIP	= *<integer>*	**1**	
}			
JRANGE			
{			
MIN	= *<integer>*	1	
MAX	= *<integer>*	**0**	
SKIP	= *<integer>*	**1**	
}			
KRANGE			
{			
MIN	= *<integer>*	1	
MAX	= *<integer>*	**0**	
SKIP	= *<integer>*	**1**	
}			
PLOTRESULTS	= *<boolean>*	**FALSE**	Indicated whether the results of the integration will be plotted in a Tecplot frame.
PLOTAS	= *<string>*	**Results**	The variable name used to plot integration results. If it contains spaces, surround it with quotes preceded by a backslash (**\'**). Ignored for forces and moments.

Range Parameters: The I-range, J-range and K-range parameters are used to limit the data altered by the equation. The specification of range indices follow the rules below.

- All indices start with one and go to some maximum index *m*.
- Zero can be used to represent the maximum index *m*; specifying zero 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 zero 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 command calculates the mass for all zones by integrating density (variable 4) over cell volumes:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'INTEGRATE  SCALARVAR = 4'
```

Example 2: Calculate the mass flux across a series of I = constant planes for zones 1, 2, and 3 and plots the results as "Mass Flux." Since the **COMMAND** string is surrounded by single quotation marks (**'**), the quotes surrounding the **PLOTAS** parameter must be preceded by a backslash to avoid a syntax error:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'INTEGRATE  [1-3]  VARIABLEOPTION =
MASSFLOWRATE INTEGRATEOVER = IPLANES  PLOTRESULTS =
TRUE  PLOTAS = \'Mass Flux\' '
```

Example 3: Calculate the "mass-weighted average" (actually the mass flow-weighted average) of total pressure, variable 7:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'INTEGRATE   [1-3]   VARIABLEOPTION =
MASSFLOWWEIGHTEDAVERAGE
SCALARVAR = 7 INTEGRATEOVER = IPLANES PLOTRESULTS =
TRUE PLOTAS = \'Mass Weighted Avg Pt\'  '
```

SAVEINTEGRATIONRESULTS

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SAVEINTEGRATOINRESULTS
FILENAME = <string>'
```

Description: Saves the most recently calculated integration results to a text file.

Required parameter:

Parameter	Syntax	Notes
FILENAME	= *<string>*	Indicates the name of the file to which to save the results. It may be a new or existing file.

Example: Save the most recent integration results to file E:\users\dave\results.txt. The backslash

characters (\) must be escaped with a second backslash character, and the file name is surrounded by quotes (“):

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SAVEINTEGRATIONRESULTS
FILENAME = "E:\\users\\dave\\results.txt"'
```

SETFIELDVARIABLES

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETFIELDVARIABLES
    [optional parameters]'
```

Description: Identifies variables in your data, such as velocity, pressure and temperature, for use in analysis.

Optional Parameters:

Parameter	Syntax	Default	Notes
CONVECTIONVARSAREMOMENTUM	= <boolean>	TRUE	Indicates whether the variables designated for Tecplot vector plots are momentum variables (density * velocity). If **FALSE**, then the vector variables must represent velocity values.
UVar	= <integer>	0	Specify the variable (by number) to use for the first Vector/Momentum variable.
VVar	= <integer>	0	Specify the variable (by number) to use for the second Vector/ Momentum variable.
WVar	= <integer>	0	Specify the variable (by number) to use for the second Vector/ Momentum variable.
ID1	= <varid>	NOTUSED	Identification of the first data set variable from which the function will be calculated.
ID2	= <varid>	NOTUSED	Identification of the second data set variable from which the function will be calculated.
VARIABLE1	= <integer>	0	Position of the first variable in the data set.
VARIABLE2	= <integer>	0	Position of the second variable in the data set.

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETFLUIDPROPERTIES [optional parameters]'
```

Description: Set the fluid properties for use by other commands.

Optional Parameters:

Parameter	Syntax	Default	Notes
INCOMPRESSIBLE	= <*boolean*>	**FALSE**	If **TRUE**, indicates an incompressible fluid.
DENSITY	= <*double*>	**1.0**	For **INCOMPRESSIBLE = TRUE**, indicates the density of the fluid.
SPECIFICHEAT	= <*double*>	**2.5**	For **INCOMPRESSIBLE = TRUE**. The value of the fluid's specific heat.
USESPECIFICHEATVAR	= <*boolean*>	**FALSE**	For **INCOMPRESSIBLE = TRUE**.
SPECIFICHEATVAR	= <*integer*>	**1**	For **INCOMPRESSIBLE = TRUE** and **USESPECIFICHEATVAR = TRUE**. The data set variable that holds the fluid's specific heat.
GASCONSTANT	= <*double*>	**1.0**	For **INCOMPRESSIBLE = FALSE**. The value of the fluid's specific gas constant.
USEGASCONSTANTVAR	= <*boolean*>	**FALSE**	For **INCOMPRESSIBLE = FALSE**.
GASCONSTANTVAR	= <*integer*>	**1**	For **INCOMPRESSIBLE = FALSE** and **USEGASCONSTANTVAR = TRUE**. The data set variable which holds the fluid's specific gas constant.
GAMMA	= <*double*>	**1.4**	For **INCOMPRESSIBLE = FALSE**. The value of the fluid's ratio of specific heats. Must be between 1 and 5/3.
USEGAMMAVAR	= <*boolean*>	**FALSE**	For **INCOMPRESSIBLE = FALSE**.
GAMMAVAR	= <*integer*>	**1**	For **INCOMPRESSIBLE = FALSE** and **USEGAMMAVAR = TRUE**. The data set variable that holds the fluid's ratio of specific heats.
VISCOSITY	= <*double*>	**1.0**	The value of the fluid's dynamic viscosity.
USEVISCOSITYVAR	= <*boolean*>	**FALSE**	
VISCOSITYVAR	= <*integer*>	**1**	For **USEVISCOSITYVAR = TRUE**. The data set variable which holds the fluid's dynamic viscosity.
CONDUCTIVITY	= <*double*>	**1.0**	The value of the fluid's conductivity.

Parameter	Syntax	Default	Notes
USECONDUCTIVITYVAR	= <boolean>	FALSE	
CONDUCTIVITYVAR	= <integer>	1	For **USECONDUCTIVITYVAR** = **TRUE**. The data set variable which holds the fluid's conductivity.

Example 1: Set the fluid properties to standard air values in meters/kilograms/seconds units:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETFLUIDPROPERTIES
   GASCONSTANT=287
   VISCOSITY=17.8E-6'
   CONDUCTIVITY=2.48E-2
```

Example 2: Set the fluid properties to incompressible with density equal to 1.0 (the default) and specific heat, viscosity and conductivity taken from data set variables 5, 6 and 7:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETFLUIDPROPERTIES
   INCOMPRESSIBLE=TRUE
   SPECIFICHEATOPTION=DATASETVAR
   SPECIFICHEATVAR=5
   VISCOSITYOPTION=DATASETVAR
   VISCOSITYVAR=6'
   CONDUCTIVITYOPTION=DATASETVAR
   CONDUCTIVITYVAR=7
```

SETGEOMETRYANDBOUNDARIES

Syntax

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'SETGEOMETRYANDBOUNDARIES
   [optional parameters]'
[RAWDATA
<boundaryrawdata>]
```

Description

Specify whether the data represent an axisymmetric flow solution (2D Cartesian plots only), whether adjacent zones should be considered to be connected at coincident faces, and specify zone boundaries and their corresponding boundary conditions.

Each line of the **RAWDATA** describes one boundary, and appears in the same format as on the Geometry and Boundaries dialog. For all boundaries, list the boundary condition and the set of zones, separated by a comma. The index range-type boundary follows this

with the boundary face, the first starting index, the first ending index, the second starting index and the second ending index. All entries are separated by commas. The boundary condition is one of **INFLOW**, **OUTFLOW**, **WALL**, **SLIPWALL**, **SYMMETRY**, **EXTRAPOLATED**. The boundary face is one of **I=1**, **I=IMAX**, **J=1**, **J=JMAX**, **K=1**, and **K=KMAX**. Refer to 19 - 4, "Setting Geometry and Boundary Options," in the *Tecplot User's Manual* for more information on boundaries.

Optional Parameters:

Parameter	Syntax	Default	Notes
AXISYMMETRIC	= *<boolean>*	**FALSE**	Can only be **TRUE** if the current plot type is 2D Cartesian. If **TRUE**, indicates that the data represents an axisymmetric solution.
SYMMETRYVAR	= *<XorY>*	**Y**	For **AXISYMMETRIC = TRUE**. Can be X or Y. Indicates which axis variable is constant along the axis of symmetry.
SYMMETRYVALUE	= *<double>*	**0.0**	For **AXISYMMETRIC = TRUE**. Indicates the value of the **SYMMETRYVAR** along the axis of symmetry.
CONNECTZONES	= *<boolean>*	**TRUE**	If **TRUE**, indicates that adjacent zones should be connected where boundary faces coincide.
NODETOLERANCE	= *<double>*	**1.0E-6**	Indicates how close two nodes must be before they will be considered coincident for the purpose of matching zone faces.
DEFAULTBC	= *<string>*	**EXTRAPOLATED**	Indicates the boundary condition that will be applied to all zone boundary faces not connected to adjacent zones or covered by zone boundaries defined by the **RAWDATA** section.

Example

Specify that the solution data represents an axisymmetric solution about X = 1. Do not allow adjacent zones to be connected. Identify two zone-type boundaries and one zone, face and index-range-type boundary:

```
$!ADDONCOMMAND
ADDONID = 'CFDAnalyzer3'
COMMAND = 'SETGEOMETRYANDBOUNDARIES
   AXISYMMETRIC = TRUE
   SYMMETRYVAR = X
   SYMMETRYVALUE = 1
   CONNECTZONES = FALSE'
RAWDATA
WALL,[2-3]
INFLOW,[4]
```


```
OUTFLOW,[1],I=IMAX,1,10,1,20
```

SETREFERENCEVALUES

Syntax:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETREFERENCEVALUES
    [optional parameters]'
```

Description: Specify the reference (free-stream) properties of the solution, identify two variables in the current data set for use with other commands.

Optional Parameters:

Parameter	Syntax	Default	Notes
RVELOCITY1ID	= <string>	**MACHNUMBER**	Identification of the first free-stream velocity component. May be **UVELOCITY** or **MACHNUMBER**.
RVELOCITY1	= <double>	**0.0**	The value of the first free-stream velocity component.
RVELOCITY2ID	= <string>	**ANGLEOFATTACK**	Identification of the second free-stream velocity component. May be **VVELOCITY** or **ANGLEOFATTACK**.
RVELOCITY2	= <double>	**0.0**	The value of the second free-stream velocity component.
RTHERMO1ID	= <string>	**DENSITY**	Identification of the first free-stream thermodynamic variable. May be **PRESSURE** or **DENSITY**.
RTHERMO1	= <double>	**1.0**	The value of the first free-stream thermodynamic variable.
RTHERMO2ID	= <string>	**SPEEDOFSOUND**	Identification of the second free-stream thermodynamic variable. May be **TEMPERATURE** or **SPEEDOFSOUND**.
RTHERMO2	= <double>	**1.0**	The value of the second free-stream thermodynamic variable.

SETUNSTEADYFLOWOPTIONS

Syntax:

```
$!ADDONCOMMAND
ADDONID = 'CFDANALYZER3'
COMMAND = 'SETUNSTEADYFLOWOPTIONS
    [SteadyState=<boolean>]
```

[RAWDATA
<timelevelrawdata>]'

Description: Identifies time levels for unsteady flow, or specifies that the solution is steady-state. If the flow is unsteady, the solution time levels are specified in the **RAWDATA** section. The first line of the **RAWDATA** section must consist of a single integer indicating the number of solution time levels. This must be followed by the time levels themselves. Each time level must be on a separate line and must consist of a floating-point number (the solution time), as well as one or more integers (the zone numbers for that solution time).

Optional Parameters:

Parameter	Syntax	Default	Notes
STEADYSTATE	= <*boolean*>	TRUE	If **TRUE**, indicates that the solution is steady-state, and the **RAWDATA**, if any, is ignored. If **FALSE**, indicates that the solution is unsteady, with time levels identified in the **RAWDATA** section.

Example: The unsteady solution contains three solution time levels of two zones each, representing solution times 0.5, 1.0 and 1.5:

```
$!ADDONCOMMAND
ADDONID = CFDAnalyzer3
COMMAND = 'SETUNSTEADYFLOWOPTIONS
   STEADYSTATE = FALSE'
RAWDATA
3
.5   1   2
1.0   3   4
1.5   5   6
```


9 - 3 Parameter Assignment Values

Parameter assignments referenced in the previous section using single angle brackets (**<>**) not defined in the *Tecplot Reference Manual*, are defined here. Note that case is not important.

Value Identifier	Allowable Values
<coeffsoption>	**GENERAL, DETAILED**
<functionname>	**IASPECTRATIO, JASPECTRATIO, KASPECTRATIO, ISTRETCHRATIO, JSTRETCHRATIO, KSTRETCHRATIO, IFACESKEWNESS, JFACESKEWNESS, KFACESKEWNESS, CELLDIAGONAL1SKEWNESS, CELLDIAGONAL2SKEWNESS, IJNORMALSSKEWNESS, JKNORMALSSKEWNESS, KINORMALSSKEWNESS, MAXNORMALSSKEWNESS, IORTHOGONALITY, JORTHOGONALITY, KORTHOGONALITY, MINORTHOGONALITY, INONPLANARITY, JNONPLANARITY, KNONPLANARITY, MINNOPLANARITY, JACOBIAN, CELLVOLUME, GRIDIUNITNORMAL, GRIDJUNITNORMAL, GRIDKUNITNORMAL, DENSITY, STAGDENSITY, PRESSURE, STAGPRESSURE, PRESSURECOEF, STAGPRESSURECOEF, PITOTPRESSURE, PITOTPRESSURERATIO, DYNAMICPRESSURE, TEMPERATURE, STAGTEMPERATURE, ENTHALPY, STAGENTHALPY, INTERNALENERGY, STAGENERGY, STAGENERGYPERUNITVOL, KINETICENERGY, UVELOCITY, VVELOCITY, WVELOCITY, VELOCITYMAG, MACHNUMBER, SPEEDOFSOUND, CROSSFLOWVELOCITY, EQUIVALENTPOTENTIALVELRAT, XMOMENTUM, YMOMENTUM, ZMOMENTUM, ENTROPY, ENTROPYMEASURES1, XVORTICITY, YVORTICITY, ZVORTICITY, VORTICITYMAG, SWIRL, VELOCITYCROSSVORTICITYMAG, HELICITY, RELATIVEHELICITY, FILTEREDRELATIVEHELICITY, SHOCK, FILTEREDSHOCK, PRESSUREGRADIENTMAG, DENSITYGRADIENTMAG, XDENSITYGRADIENT, YDENSITYGRADIENT, ZDENSITYGRADIENT, SHADOWGRAPH, DIVERGENCEOFVELOCITY, SUTHERLANDSLAW, ISENTROPICDENSRAT, ISENTROPICPRESRAT, ISENTROPICTEMPRAT, VELOCITY, VORTICITY, MOMENTUM, PERTURBATIONVELOCITY, VELOCITYCROSSVORTICITY, PRESSUREGRADIENT, DENSITYGRADIENT, VELOCITYGRADIENT**
<gravitydirection>	MINUSX, MINUSY, MINUSZ, PLUSX, PLUSY, PLUSZ
<integrationoption>	**CELLVOLUMES, IPLANES, JPLANES, KPLANES, ILINES, JLINES, KLINES**
<normalizationoption>	**NONE, MAXIMUMMAGNITUDE, REFERENCEVALUES**
<particlefunction>	**PARTICLEPATH, STREAKLINE**
<releaseoption>	**TIMELEVEL, UNITTIME**
<specifyoption>	**SPECIFY, CALCULATE**
<storeoption>	**PARTICLEVALUES, FLUIDVALUES**
<terminationoption>	**TEMPERATURE, ABLATE**
<turbulencefunction>	**ENERGY, DISSIPATIONRATE, FREQUENCY, VISCOSITY**
<variableoption>	**LENGTHAREAVOLUME, SCALAR, AVERAGE, MASSWEIGHTEDSCALAR, MASSWEIGHTEDAVERAGE, WEIGHTEDAVERAGE, SCALARFLOWRATE, MASSFLOWRATE, MASSWEIGHTEDFLOWRATE, MASSFLOWWEIGHTEDAVERAGE, FORCESANDMOMENTS, VECTORDOTNORMAL, VECTORAVERAGE, VECTORDOTTANGENTIAL**
<varid>	**PRESSURE, TEMPERATURE, DENSITY, STAGNATIONENERGY, MACHNUMBER, NOTUSED**
<XorY>	**X, Y**

Chapter 10 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 11, “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>*	
USELIGHTSOURCETOFIL	= *<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	<<gridlinedetail>>	
MINORGRIDLINES	<<gridlinedetail>>	
TICKS	<<ticklabeldetail>>	
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		Default	Notes
{			
SHOW	= <*boolean*>		
SHOWBOTHDIRECTIONS	= <*boolean*>	FALSE	Non-3D only.
SHOWPERPENDICULAR	= <*boolean*>	FALSE	Non-3D only.
SHOWOPPOSITEEDGE	= <*boolean*>	FALSE	3D only
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		Default	Notes
{			
SHOWONAXISLINE	= <*boolean*>	TRUE	
SHOWONGRIDBORDERMIN	= <*boolean*>	FALSE	Non-3D only.
SHOWONGRIDBORDERMAX	= <*boolean*>	FALSE	Non-3D only.
SHOWONOPPOSITEEDGE	= <*boolean*>	FALSE	3D only.
SHOWONALLAXES	= <*boolean*>	TRUE	Polar R only.
SHOWONVIEWPORTTOP	= <*boolean*>	TRUE	Polar only.
SHOWONVIEWPORTBOTTOM	= <*boolean*>	TRUE	Polar only.
SHOWONVIEWPORTLEFT	= <*boolean*>	TRUE	Polar only.
SHOWONVIEWPORTRIGHT	= <*boolean*>	TRUE	Polar only.
TITLEMODE	= <*axistitlemode*>		
TEXT	= <*string*>		
COLOR	= <*color*>		
TEXTSHAPE	<<textshape>>		
OFFSET	= <*double*>		
PERCENTALONGLINE	= <*double*>	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	= <double>	
CMAX	= <double>	

Example: Set the continuous color.

```
$!GLOBALCONTOUR VAR = 4
$!FIELDLAYERS SHOWCONTOUR = YES

$!GLOBALCONTOUR COLORMAPFILTER
  {COLORMAPDISTRIBUTION = CONTINUOUS}
$!GLOBALCONTOUR COLORMAPFILTER
  {
   CONTINUOUSCOLOR
    {
     CMIN = 0.5
     CMAX = 2
    }
  }
}
```

<<dialogplacement>>

Description: Describes the placement for a dialog.

Expands to:

Syntax		Notes
{		
ANCHORALIGNMENT	= <anchoralignment>	
ANCHORHORIZONTALINSIDE	= <boolean>	ANCHORHORIZONTALINSIDE and ANCHORVERTICALINSIDE control how the dialog window is anchored in both the horizontal and vertical directions relative to the Tecplot main window.
ANCHORVERTICALINSIDE	= <boolean>	
MINVISIBILITYPERCENTAGE	= <integer>	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.
XOFFSET	= <integer>	XOFFSET and YOFFSET are in pixels. They may be negative, but will be truncated to the bounding rectangle of the Tecplot main window.
YOFFSET	= <integer>	
POSITIONATANCHOR	= <positionatanchor>	POSITIONATANCHOR 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
 }
}
```

<<gridarea>>

Description: Change settings for the axis grid area.

Expands to:

Syntax		Notes
{		
DRAWGRIDLAST	= *<boolean>*	
DRAWBORDER	= *<boolean>*	Not available in 3D.
LINETHICKNESS	*<op>* *<dexp>*	
COLOR	= *<color>*	
ISFILLED	= *<boolean>*	
FILLCOLOR	= *<color>*	
USELIGHTSOURCETOFILL	= *<boolean>*	Only available for 3D.
LABELSALLSIDES	= *<boolean>*	
TICKSALLSIDES	= *<boolean>*	
EXTENTS	*<<rect>>*	Not available in 3D.
}		

Example: Turn on the grid area border for a 2-D plot and change the line thickness to be 2 percent:

```
$!TWODAXIS
   GRIDAREA
   {
    DRAWBORDER = YES
    LINETHICKNESS = 2
   }
```

<<gridlinedetail>>

Description: Change settings for axis gridlines.

Expands to:

Syntax		Notes
{		
COLOR	= *<color>*	
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 J K }	<op> <integer> <op> <integer> <op> <integer>	

Example: Set the I- and J-index skip for vectors to 2 for all zones:

```
$!FIELDMAP
VECTOR
{
 IJKSKIP
 {
 I = 2
 J = 2
 }
}
```

<<indexrange>>

Description: Set an index range.

Expands to:

Syntax		Notes
{ MIN MAX SKIP }	 <op> <integer> <op> <integer> <op> <integer>	

Example: Change the plot so the data set shows I-planes 3, 5, and 7 for zones 1 to 3:

```
$!FIELDMAP [1-3]
SURFACES
{
 SURFACESTOPLOT = IPLANES
 IRANGE
 {
   MIN = 3
   MAX = 7
   SKIP = 2
 }
}
```

<<numberformat>>

Description: Set the format used to draw a number.

Expands to:

Syntax		Default	Notes
{			
FORMATTING	= <valueformat>		
CUSTOMLABEL	= <integer>		
DYNAMICLABELNAME	= <string>		Name of the dynamic label generator to use when "Formatting" is set to "DynamicLabel"
PRECISION	<op> <integer>		
SHOWDECIMALSONWHOLENUMBERS	= <boolean>	FALSE	
REMOVELEADINGZEROS	= <boolean>	FALSE	
SHOWNEGATIVESIGN	= <boolean>	*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
{		All values are in inches.
WIDTH	<op> <dexp>	
HEIGHT	<op> <dexp>	
LEFTHARDCLIPOFFSET	<op> <dexp>	
RIGHTHARDCLIPOFFSET	<op> <dexp>	
TOPHARDCLIPOFFSET	<op> <dexp>	
BOTTOMHARDCLIPOFFSET	<op> <dexp>	
}		

Example: Change the left hardclip offset for **LETTER** size paper to be 0.25 inches:

```
$!PAPER
PAPERSIZEINFO
{
 LETTER
 {
  LEFTHARDCLIPOFFSET = 0.25
 }
}
```

<<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
{		Units are based on the parent command.
X1	<op> <dexp>	
Y1	<op> <dexp>	
X2	<op> <dexp>	
Y2	<op> <dexp>	
}		

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	<<xyz>>	
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>*	If thin patterned lines are not drawn correctly, set **STIPPLEALLLINES** to **ALL**.
DEPTHBUFFERSIZE	= *<integer>*	For low memory graphics cards, the depth buffer size may need to be reduced.
MINBITSPERRGBPLANE	= *<integer>*	Specify the minimum number of bits used for each of the planes in the image buffer.
DOEXTRADRAWFORLASTPIXEL	= *<boolean>*	Sometimes the last pixel for stroked font characters is not drawn If so, turn **DOEXTRADRAWFORLASTPIXEL** on.
MAXSTRIPLENGTH	= *<integer>*	Some graphics cards have problems with long strips. Use **MAXSTRIPLENGTH** to reduce the strip length.
MAXPRIMATIVESPERBLOCK	= *<integer>*	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.
CONSTANTLYUSESCISSORING	= *<boolean>*	Turn **ConstantlyUseScissoring** on if you see lines extending outside the borders of the frame. There is a slight performance penalty when using this option.
USEQUADSTRIPS	= *<boolean>*	If some shaded or contour flooded quads or triangles do not appear or are black, try turning this off.
USETRIANGLESTRIPS	= *<boolean>*	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.
TRIANGULATEFILLEDPOLYGONS	= *<boolean>*	As with **USEQUADSTRIPS**, try turning on **TRIANGULATEFILLEDPOLYGONS** if you are still experiencing problems even after turning off **USETRIANGLESTRIPS** and **USEQUADSTRIPS**.
USEGLCOLORMATERIALFUNCTION	= *<boolean>*	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.
MAXTEXTURESIZE	= *<integer>*	
FORCESMOOTHSHADINGFORLIGHTING	= *<boolean>*	
ADJUSTRECTANGLERIGHTANDBOTTOM	= *<boolean>*	
}		

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
{		Shade values can range from 0 (black) to 100 (white).
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*>	
}		

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		
{		
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 <<textbox>>.

<<textshape>>

Description: Change settings related to text font and character height.

Expands to:

Syntax		Notes
{ FONT SIZEUNITS HEIGHT }	= *<font>* = *<sizeunits>* *<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		Default	Notes
{			
SHOWONAXISLINE	= <*boolean*>	TRUE	
SHOWONGRIDBORDERMIN	= <*boolean*>	FALSE	Non-3D only.
SHOWONGRIDBORDERMAX	= <*boolean*>	FALSE	Non-3D only.
SHOWONOPPOSITEEDGE	= <*boolean*>	FALSE	3D only.
SHOWONALLAXES	= <*boolean*>	TRUE	Polar R only.
SHOWATAXISINTERSECTION	= <*boolean*>		
SKIP	= <*integer*>		
ERASEBEHINDLABELS	= <*boolean*>		
NUMFORMAT	<<*numberformat*>>		
TEXTSHAPE	<<*textshape*>>		Not allowed to change size units parameter.
OFFSET	<*op*> <*dexp*>		
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		Default	Notes
{			
SHOWONAXISLINE	= <boolean>	**TRUE**	
SHOWONGRIDBORDERMIN	= <boolean>	FALSE	Non-3D only.
SHOWONGRIDBORDERMAX	= <boolean>	FALSE	Non-3D only.
SHOWONOPPOSITEEDGE	= <boolean>	FALSE	3D only.
SHOWONALLAXES	= <boolean>	TRUE	Polar R only.
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 11

Parameter Assignment Values, Expressions, and Arithmetic and Logical Operators

11 - 1 Assignment Value Table

Parameter assignments referenced in the previous chapters using single angle brackets (< >) are defined here. (Case is not important.)

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<*altmousebuttonmode*>	REDRAW, REVERTTOSELECT
<*addonstyle*>	V7STANDARD,V7ACTIVEX
<*anchoralignment*>	TOPLEFT, TOPCENTER, TOPRIGHT, MIDDLELEFT, MIDDLECENTER, MIDDLERIGHT, BOTTOMLEFT, BOTTOMCENTER, BOTTOMRIGHT
<*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
<*borderlocation*>	IBORDER, JBORDER, KBORDER
<*boundarycondition*>	FIXED, ZEROGRADIENT, ZERO2ND
<*boxtype*>	NONE, FILLED, HOLLOW

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<charactersequence>	One or more printable characters.
<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
<compressiontype>	BESTSPEED, SMALLESTSIZE
<conditionalexp>	*<dexp>* *<relop>* *<dexp>* or *<string>* *<relop>* *<string>*.
<constrainintop2mode>	USEVAR, USECONSTANT
<contourcoloring>	RGB, GROUP1, GROUP2, GROUP3, GROUP4
<contourlabelaction>	ADD, DELETEALL
<contourlabellocation>	CONTOURLEVELS, INCREMENT, COLORMAPDIVISIONS
<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
<dataloadstrategy>	AUTO, HEAP
<derivpos>	SIMPLE, ATPOINT, COMPLEX, ATPOINTB2

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<dialogname>	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
<derivpos>	SIMPLE, ATPOINT, COMPLEX
<dexp>	*<double>*, ((*<expression>*))
<double>	Valid floating point value.
<draworder>	BEFOREDATA, AFTERDATA
<drift>	NONE, LINEAR, QUAD
<edgesetting>	NONE, MIN, MAX, BOTH
<edgetype>	BORDERS, CREASES, BORDERSANDCREASES
<epspreviewimagetype>	NONE, TIFF, EPSIV2, FRAME

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<errorbartype>	UP, DOWN, LEFT, RIGHT, VERT, HORZ, CROSS
<exportformat>	RASTERMETAFILE, TIFF, SUNRASTER, XWINDOWS, PSIMAGE, PS, EPS, WINDOWSMETAFILE, BMP, PNG, AVI, JPEG
<expression>	See Figure 11 - 2, “Assignment Value Expressions,” on page 280.
<fielddatatype>	FLOAT, DOUBLE
<fillmode>	NONE, USESPECIFICCOLOR, USEBACKGROUNDCOLOR, USELINECOLOR
<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`**
<imagetype>	LOSSLESS, JPEG, 256COLOR
<integer>	integer constants or variables containing an integer. **NOTE** expressions that logically result in integer are not currently supported.
<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 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<*macrointrinsic*>	AXISMAXX, AXISMAXY, AXISMAXZ, AXISMINX, AXISMINY, AXISMINZ, COLORMAPDYNAMIC, ENDSLICEPOS, FRAMEMODE, IS3DV, LOOP, MACROFILEPATH, MAXB, MAXC, MAXI, MAXJ, MAXK, MAXS, MAXU, MAXV, MAXV*nn*, MAXW, MAXX, MAXY, MAXZ, MINB, MINC, MINS, MINU, MINV, MINV*nn*, MINW, MINX, MINY, MINZ, NUMFRAMES, NUMPLANES, NUMVARS, NUMWIN, NUMXYMAPS, NUMZONES, OPSYS, PLATFORM, SLICEPLANETYPE, SOLUTIONTIME, STARTSLICEPOS, TECHOME, TECPLOTVERSION
<*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
<*pointselection*>	ALLPOINTS, NEARESTNPOINTS, OCTANTNPOINTS

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<pointstoplot>	SURFACESONLY, ALL
<positionatanchor>	ONCE,NEVER,ALWAYS
<printerdriver>	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, BOXFILTER, TRIANGLEFILTER, BELLFILTER, BSPLINEFILTER, CUBICFILTER, MITCHELFILTER, GAUSSIANFILTER
<rgblegendorientation>	RGB, GBR, BRG, RBG, BGR, GRB
<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>*]
<shadetype>	SOLIDCOLOR, PANELED, GOURAUD, COLOREDPANELED,
<sidebarsizing>	MAXOFALL, DYNAMIC
<sizeunits>	GRID, FRAME, POINT
<skipmode>	BYINDEX, BYFRAMEUNITS
<slicesource>	VOLUMEZONES, SURFACEZONES, SURFACESOFVOLUMEZONES, LINEARZONES
<slicesurface>	XPLANES, YPLANES, ZPLANES, IPLANES, JPLANES, KPLANES
<sortby>	NONE, BYDEPENDENDTVAR, BYINDEPENDENTVAR, BYSPECIFICVAR
<specifyrgb>	SPECIFYRGB, SPECIFYRG, SPECIFYRB, SPECIFYGB
<spherescatterrenderquality>	LOW, MEDIUM, HIGH.
<standardcolormap>	SMRAINBOW, LGRAINBOW, MODERN, GRAYSCALE, TWOCOLOR
<stipplemode>	ALL, CRITICAL, NONE
<streamdirection>	FORWARD, REVERSE, BOTH
<streamtype>	SURFACELINE, VOLUMELINE, VOLUMERIBBON, VOLUMEROD, TWODLINE

Table 11-1. Parameter Assignment Values.

Value Identifier	Allowable Values
<*string*>	"<*charactersequence*>", '<*charactersequence*>'[b]
<*stylebase*>	FACTORY, CONFIG
<*subboundary*>	ADD, ADDONLY, ALL, REMOVE
<*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
<*threedviewchangedrawlevel*>	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
<*unloadstrategy*>	AUTO, NEVERUNLOAD, MINIMIZEMEMORYUSE
<*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).

11 - 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 11-1. 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.

Table 11-1. Functions supported by Tecplot.

sqrt(*x*)	Square root of x.
tan(*x*)	Tangent of x in radians.
tanh(*x*)	Hyperbolic tangent of x.

Constants are also supported, as listed in the following table.

Table 11-2. 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 11-3. 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 12 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 13 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 $!FIELDMAP and **$!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

Index

Numerics

A

D

F

G

I

M

N

Z