APDL Programmer's Guide

ANSYS Release 8.1

001973
April 2004

APDL Programmer's Guide

ANSYS Release 8.1

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Revision History

Number	Release	Date
001620	ANSYS 6.1	March 2002
001695*	ANSYS 7.0	October 2002
001788*	ANSYS 7.1	May 2003
001901*	ANSYS 8.0	October 2003
001973**	ANSYS 8.1	April 2004

* ANSYS Documentation on CD.

** Included in ANSYS Documentation on CD and in print.

Trademark Information

Table of Contents

List of Figures

List of Tables

Chapter 1: Introducing APDL

1.1. What Is APDL?

APDL stands for ANSYS Parametric Design Language, a scripting language that you can use to automate common tasks or even build your model in terms of parameters (variables). APDL also encompasses a wide range of other features such as repeating a command, macros, if-then-else branching, do-loops, and scalar, vector and matrix operations.

While APDL is the foundation for sophisticated features such as design optimization and adaptive meshing, it also offers many conveniences that you can use in your day-to-day analyses. In this guide we'll introduce you to the basic features - parameters; macros; branching, looping, and repeating; and array parameters - and show you some simple examples. As you become more adept at the language, you will see that the applications for APDL are limited only by your imagination.

Chapter 2: Working with the Toolbar

2.1. Adding Commands to the Toolbar

You can add frequently used ANSYS functions or macros to the ANSYS toolbar (creating macros is covered starting in Chapter 4, "APDL as a Macro Language"). You do this by defining *abbreviations*. An abbreviation is an alias (up to eight characters long) for an ANSYS command, GUI function name, or macro name. For example, MATPROP might be an abbreviation for a macro that lists material properties, SAVE_DB is an abbreviation for the **SAVE** command, and **QUIT** is an abbreviation for the **Fnc_/EXIT** function (which launches the **Exit from ANSYS** dialog box).

The ANSYS program provides two ways to use abbreviations. You can issue the abbreviation (and execute the macro, command, etc. that it performs) by typing it at the beginning of a command line. If you are using the ANSYS GUI, you can also execute the macro or command by pressing the appropriate button on the ANSYS toolbar.

The toolbar shown in Figure 2.1: "Toolbar" contains buttons that correspond to existing abbreviations.

Figure 2.1 Toolbar

While some abbreviations, such as SAVE_DB, are predefined, the abbreviations the toolbar contains and the functions they execute are up to you. A single toolbar can hold up to 100 abbreviations (you can "nest" toolbars to extend this number). You can redefine or delete abbreviations at will; however, abbreviations are not automatically saved and must be explicitly saved to a file and reloaded for each ANSYS session.

2.2. Modifying the Toolbar

You can create abbreviations either through the ***ABBR** command or through the **Utility Menu> Macro> Edit Abbreviations** or **Utility Menu> MenuCtrls> Edit Toolbar** menu items. Using one of the menu items is preferable for two reasons:

- Clicking OK automatically updates the toolbar (using the ***ABBR** command requires that you use the **Utility Menu> MenuCtrls> Update Toolbar** menu item to make your new abbreviation appear on the toolbar).
- You can easily edit the abbreviation if required.

The syntax for the ***ABBR** command and related dialogs is

***ABBR**, *Abbr*, *String*

Abbr
The abbreviation name that will appear on the toolbar button. The name can contain up to eight characters.

String
The *String* argument is the name of the macro or command that *Abbr* represents. If *String* is the name of a macro, the macro must be within the macro search path. For more information about using macros, see Chapter 4, "APDL as a Macro Language". If *String* references an ANSYS picking menu or dialog box (using

UIDL), then specify "Fnc_string." For example, in the abbreviation definitions for "QUIT" and "POWRGRPH" shown above, "Fnc_/QUIT" and "Fnc_/GRAPHICS" are unique UIDL function names which identify the ANSYS picking menu or dialog box associated with the QUIT and POWRGRPH abbreviations respectively. For more information about accessing UIDL functions, see Section 5.6: Calling Dialog Boxes From a Macro. *String* can contain up to 60 characters but cannot include any of the following:

- The character "$"
- The commands **C***, **/COM**, **/GOPR**, **/NOPR**, **/QUIT**, **/UI**, or ***END**

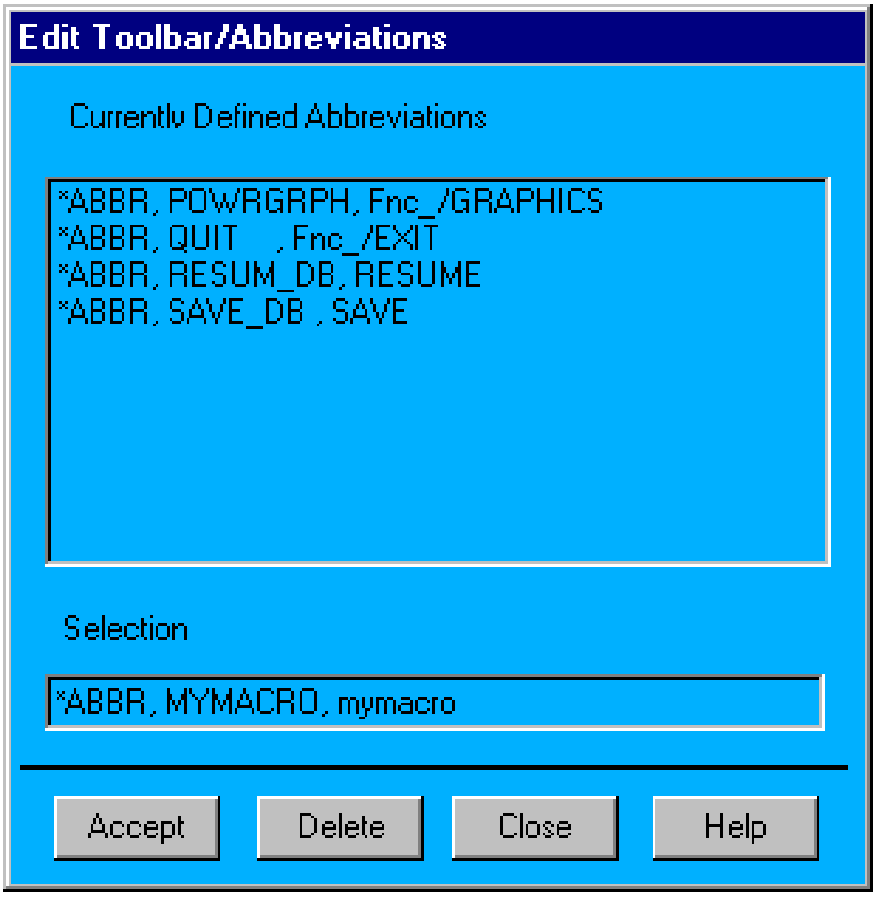
The default ANSYS toolbar has the following abbreviations predefined:

```
*ABBR, SAVE_DB, SAVE
*ABBR, RESUM_DB, RESUME
*ABBR, QUIT, Fnc_/EXIT
*ABBR, POWRGRPH, Fnc_/GRAPHICS
```

2.2.1. Example: Adding a Toolbar Button

For example, to add a button to the toolbar that calls the macro file **mymacro.mac**, you would enter the values shown in the following figure in the **Utility Menu> MenuCtrls> Edit Toolbar** dialog box.

Figure 2.2 Adding a New Abbreviation



The new button is appended to the button bar as shown in the following figure.

Figure 2.3 Toolbar with New Button

2.2.2. Saving Toolbar Buttons

Toolbar buttons are not persistent from one ANSYS session to the next; however, they are saved and maintained in the database so that any "resume" of the session will still contain these abbreviations. To save your custom button definitions, you must explicitly save them to a file through the **Utility Menu> MenuCtrls> Save Toolbar** menu item (**ABBSAV** command) and restore them for each session using the **Utility Menu> MenuCtrls> Restore Toolbar** menu item (**ABBRES** command). You can do this programmatically in a macro.

Note — If any abbreviations already exist in the named file, the **ABBSAV** command overwrites them.

The format of the abbreviations file is the APDL commands that are used to create the abbreviations. Thus, if you wish to edit a large set of buttons or change their order, you may find using a text editor to be the most convenient method. For example, the following is the file that results from saving the default toolbar buttons.

```
/NOPR
*ABB,SAVE_DB ,SAVE
*ABB,RESUM_DB,RESUME
*ABB,QUIT    ,Fnc_/EXIT
*ABB,POWRGRPH,Fnc_/GRAPHICS
/GO
```

The ***ABB** commands (the abbreviated form of ***ABBR**) define the buttons. The **/NOPR** at the top turns off echoing to the log file while the **/GO** at the bottom turns log file echoing on.

2.3. Nesting Toolbar Abbreviations

The save-and-restore features described above allow you to *nest* abbreviations. By nesting abbreviations under one button, you can define specialized toolbars (if you have many abbreviations, having them on a single toolbar can be cluttered, making it difficult to find the proper button). To nest abbreviations, you simply define an abbreviation that restores an abbreviation file. For example, the following command defines PREP_ABR as an abbreviation that restores abbreviations from the file **prep.abbr**.

```
*ABBR,PREP_ABR,ABBRES,,PREP,ABBR
```

PREP_ABR will appear as a button on the toolbar. Clicking it will replace the existing buttons with the set of buttons defined in the **prep.abbr** file.

By defining abbreviations to restore these files and including those abbreviations in the appropriate files, you can have a virtually unlimited number of abbreviations in a given ANSYS session. You can even extend this concept and create your own menu hierarchy by nesting several abbreviation files. If you implement such a hierarchy, it's a good practice to add an abbreviation as a "return" button in each file to navigate back through the menus.

Chapter 3: Using Parameters

3.1. Parameters

Parameters are APDL variables (they are more similar to Fortran variables than to Fortran parameters). You don't need to explicitly declare the parameter type. All numeric values (whether integer or real) are stored as double-precision values. Parameters that are used but not defined are assigned a near-zero, or "tiny," value of approximately 2^{-100}. For example, if parameter A is defined as A=B, and B is not defined, then A is assigned the tiny value.

ANSYS uses two types of parameters: scalar and array. The first part of this chapter discusses information that is applicable to both types. Starting with Section 3.11: Array Parameters, the information is specific to array type parameters.

Character strings (up to eight characters long) can be assigned to parameters by simply enclosing the string in single quotes. APDL also provides several types of array parameters: numeric, character, string and table (a special numeric type that automatically interpolates values).

You can use a parameter (instead of a literal number or character string) as an argument to any ANSYS command; the parameter is evaluated and its current value is used for that argument. For example, if you assign the value 2.7 to a parameter named AA and then issue the command

```
N,12,AA,4
```

the ANSYS program will interpret the command as

```
N,12,2.7,4
```

(which defines node 12 at X=2.7 and Y=4).

> *Note* — If array, table, or character parameters are used within a macro or input file, those parameters should be dimensioned (if array or table) and defined within that macro or input file. If you fail to follow this practice, ANSYS will produce error messages stating that those parameters are undefined. ANSYS will produce the error messages even if the parameters lie within unexecuted ***IF** statements, as parameter substitution is done before the branching for the ***IF** is checked.

3.2. Guidelines for Parameter Names

Parameter names must:

- Begin with a letter
- Contain only letters, numbers, and underscore characters
- Contain no more than 32 characters

Examples of valid and invalid parameter names are

Valid:

ABC
PI
X_OR_Y

Invalid:

MY_PARAMETER_NAME_LONGER_THAN_32_CHARACTERS (more than 32 characters)
2CF3 (begins with a number)
M&E (invalid character "&")

When naming parameters:

- Avoid parameter names that match commonly used ANSYS labels, such as:
  - Degree of freedom (DOF) labels (TEMP, UX, PRES, etc.)
  - Convenience labels (ALL, PICK, STAT, etc.)
  - User-defined labels (such as those defined with the **ETABLE** command)
  - Array type field labels (such as CHAR, ARRAY, TABLE, etc.)
- Parameter names ARG1 through ARG9 and AR10 through AR99 are reserved for local parameters. Generally, local parameters are used in macros (see Section 4.4: Local Variables). Use of these names as "regular" parameters is not recommended.
- Parameter names must not match abbreviations defined with the ***ABBR** command. For more information about abbreviations, see Section 2.1: Adding Commands to the Toolbar.
- Do not begin parameter names with an underscore (_). This convention is reserved for parameters used by the GUI and ANSYS-supplied macros.
- APDL programmers supporting an organization should consider naming their parameters with a trailing underscore(_). These can displayed as a group using the ***STATUS** command and deleted from memory as a group through the ***DEL** command.

3.2.1. Hiding Parameters from *STATUS

Section 3.3.4: Listing Parameters discusses listing parameters through the ***STATUS** command. You can use a parameter naming convention to "hide" parameters from the ***STATUS** command. Any parameter whose name ends in an underscore (_) will not be listed by ***STATUS**.

This capability was added specifically for those who are developing APDL macros for large audiences. You can use this to build macros that your ANSYS users and other macro programmers cannot list.

3.3. Defining Parameters

Unless otherwise specified, the information in the next several sections applies to both scalar and array type parameters. Beginning with Section 3.11: Array Parameters, the information is specific to array type parameters.

You can either assign values to parameters or retrieve values supplied by ANSYS and store these values in parameters. For retrieving values from ANSYS, you can use either the ***GET** command or the various in-line get functions. The following sections cover these subjects in detail.

3.3.1. Assigning Parameter Values During Execution

You can use the ***SET** command to define parameters. The following examples illustrate a set of example parameters defined using ***SET**:

```
*SET,ABC,-24
*SET,QR,2.07E11
```

```
*SET,XORY,ABC
*SET,CPARM,'CASE1'
```

You can use an "=" as a shorthand way of calling the ***SET** command (this is the most convenient method). The format of the shortcut is *Name* = *Value*, where *Name* is the name assigned to the parameter and *Value* is the numeric or character value stored in that parameter. For character parameters, the assigned value must be enclosed in single quotes and cannot exceed eight alphanumeric characters. The following are examples of "=" in use:

```
ABC=-24
QR=2.07E11
XORY=ABC
CPARM='CASE1'
```

In the GUI, you can either type the "=" directly in the ANSYS input window or in the "Selection" field of the **Scalar Parameter** dialog box (accessed by the **Utility Menu> Parameters> Scalar Parameters** menu item).

3.3.2. Assigning Parameter Values At Startup

You can define parameters as arguments when launching ANSYS from the operating system command line. Simply type parameter definitions after the ANSYS execution command (which is system dependent) using the format *-Name Value*. For example, the following defines two parameters (parm1 and parm2) having the values 89.3 and -0.1:

```
ansys81 -parm1 89.3 -parm2 -0.1
```

It is a good practice to avoid assigning one or two character parameter names at startup to avoid conflicts with ANSYS command line options.

> *Note* — Remember that UNIX shells treat single quotes and many other non-alphanumeric characters as special symbols. When defining character parameters, you must tell UNIX not to interpret the quotes by inserting a back slash (\) before the single quotes. For example, the following defines two character parameters having the values `filename' and `200.'
>
> ```
> ansys81 -cparm1 \'filename\' -cparm2 \'200\'
> ```

If you use the ANSYS Launcher to start ANSYS, you can define parameters through the **Interactive** or **Batch** menu items (using the *-Name Value* format described above).

If you are defining a large number of parameters at startup, you'll find it much more convenient to define these in the **start81.ans** file or through a separate file that you can load through the **/INPUT** command instead of the command line.

3.3.3. Assigning ANSYS-Supplied Values to Parameters

ANSYS provides two powerful methods for retrieving values:

- The ***GET** command, which retrieves a value from a specified item and stores it in a specified parameter.
- The in-line get functions, which can be used in operations. Each get function returns a specific value from a specific item.

3.3.3.1. Using the *GET Command

The ***GET** command (**Utility Menu> Parameters> Get Scalar Data**) retrieves an ANSYS-supplied value for an item (a node, an element, an area, etc.) and stores it as a user-named parameter. Various keyword, label, and number combinations identify the retrieved item. For example, ***GET**,A,ELEM,5,CENT,X returns the centroid x-location of element 5 and stores the result as parameter A.

The format for the ***GET** command is:

```
*GET,Par,Entity,ENTNUM,Item1,IT1NUM,Item2,IT2NUM
```

where

- *Par* is the name of the parameter to store the retrieved item.
- *Entity* is a keyword for the item to be stored. Valid keywords are NODE, ELEM, KP, LINE, AREA, VOLU, etc. For a complete list of valid keywords, see the ***GET** description in the *ANSYS Commands Reference*.
- *ENTNUM* is the number of the entity (or zero for all entities).
- *Item1* is the name of an item for a particular entity. For example, if *Entity* is ELEM, *Item1* will be either NUM (the highest or lowest element number in the selected set) or COUNT (the number of elements in the set). (For a complete list of *Item1* values for each entity type, see the ***GET** description in the *ANSYS Commands Reference*.)

You can think of the ***GET** command as a path down a tree structure, from general to specific information.

The following examples show the ***GET** command in use. The first command below *gets* the material attribute (the MAT reference number) of element 97 and assigns it to parameter BCD:

```
*GET,BCD,ELEM,97,ATTR,MAT     ! BCD = Material number of element 97
*GET,V37,ELEM,37,VOLU         ! V37 = volume of element 37
*GET,EL52,ELEM,52,HGEN        ! EL52 = value of heat generation in element 52
*GET,OPER,ELEM,102,HCOE,2     ! OPER = heat coefficient of element 102,face2
*GET,TMP,ELEM,16,TBULK,3      ! TMP = bulk temperature of element 16,face3
*GET,NMAX,NODE,,NUM,MAX       ! NMAX = maximum active node number
*GET,HNOD,NODE,12,HGEN        ! HNOD = value of heat generation at node 12
*GET,COORD,ACTIVE,,CSYS       ! COORD = active coordinate system number
```

3.3.3.2. Using In-line Get Functions

For some items, you can use in-line "get functions" in place of the***GET** command. A get function returns a value for an item and uses it directly in the current operation. This process allows you to bypass the dual steps of storing the value with a parameter name and then entering the parameter name in an operation. For example, suppose that you want to calculate the average x-location of two nodes. You could do the following using the ***GET** function:

1. Issue the following command to assign the x-location of Node 1 to parameter L1.

   ```
   *GET,L1,NODE,1,LOC,X
   ```

2. Issue a second ***GET** command to assign the x-location of Node 2 to parameter L2.
3. Compute the middle location from MID=(L1+L2)/2.

A shorter method is to use the node location "get function" NX(*N*), which returns the x-location of node *N*. You can use it to calculate the MID location without setting intermediate parameters L1 and L2, as is shown in the following example:

```
MID=(NX(1)+NX(2))/2
```

Get function arguments can themselves be parameters or other get functions. For instance, get function NELEM(ENUM,NPOS) returns the node number in position NPOS for element ENUM. Combining functions NX(NELEM(ENUM,NPOS)) returns the x-location of that node.

The following table summarizes the available get functions:

Get Function	Retrieved Value
Entity Status:	
NSEL(*N*)	Status of node *N* (-1=unselected, 0=undefined, 1=selected)
ESEL(*E*)	Status of element *E* (-1=unselected, 0=undefined, 1=selected)
KSEL(*K*)	Status of keypoint *K*(-1=unselected, 0=undefined, 1=selected)
LSEL(*L*)	Status of line *L*(-1=unselected, 0=undefined, 1=selected)
ASEL(*A*)	Status of area *A* (-1=unselected, 0=undefined, 1=selected)
VSEL(*V*)	Status of volume *E* (-1=unselected, 0=undefined, 1=selected)

Next Selected Entity:	
NDNEXT(*N*)	Next selected node having a node number greater than *N*
ELNEXT(*E*)	Next selected element having an element number greater than *E*
KPNEXT(*K*)	Next selected keypoint having a keypoint number greater than *K*
LSNEXT(*L*)	Next selected line having a line number greater than *L*
ARNEXT(*A*)	Next selected area having an area number greater than *A*
VLNEXT(*V*)	Next selected volume having a volume number greater than *V*

Locations:	
CENTRX(*E*)	Centroid x-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
CENTRY(*E*)	Centroid y-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
CENTRZ(*E*)	Centroid z-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
NX(*N*)	X-coordinate of node *N* in the active coordinate system
NY(*N*)	Y-coordinate of node *N* in the active coordinate system
NZ(*N*)	Z-coordinate of node *N* in the active coordinate system
KX(*K*)	X-coordinate of keypoint *K* in the active coordinate system
KY(*K*)	Y-coordinate of keypoint *K* in the active coordinate system
KZ(*K*)	Z-coordinate of keypoint *K* in the active coordinate system
LX(*L*, *LFRAC*)	X-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0)
LY(*L*, *LFRAC*)	Y-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0)
LZ(*L*, *LFRAC*)	Z-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0)

Nearest to Location:	
NODE(*X*, *Y*, *Z*)	Number of the selected node nearest the *X*, *Y*, *Z* point (in the active coordinate system; lowest number for coincident nodes)
KP(*X*, *Y*, *Z*)	Number of the selected keypoint nearest the *X*, *Y*, *Z* point (in the active coordinate system; lowest number for coincident keypoints)

Distance:	
DISTND(*N1*,*N2*)	Distance between nodes*N1* and *N2*
DISTKP(*K1*, *K2*)	Distance between keypoints *K1* and *K2*
DISTEN(*E*, *N*)	Distance between the centroid of element *E* and node *N*. Centroid is determined from the selected nodes on the element.

Angles:	
ANGLEN(*N1, N2, N3*)	Subtended angle between two lines (defined by three nodes where *N1* is the vertex node). Default is in radians (see the ***AFUN** command to select degrees).
ANGLEK(*K1, K2, K3*)	Subtended angle between two lines (defined by three keypoints where *K1* is the vertex keypoint). Default is in radians (see the ***AFUN** command to select degrees).

Nearest to Entity:	
NNEAR(*N*)	Selected node nearest node *N*
KNEAR(*K*)	Selected keypoint nearest keypoint *K*
ENEARN(*N*)	Selected element nearest node *N*. The element position is calculated from the selected nodes.

Areas:	
AREAND(*N1, N2, N3*)	Area of the triangle with vertices at nodes *N1, N2, N3*
AREAKP(*K1, K2, K3*)	Area of the triangle with vertices at keypoints *K1, K2, K3*
ARNODE(*N*)	Area at node *N* apportioned from selected elements attached to node *N*. For 2-D planar solids, returns edge area associated with the node. For axisymmetric solids, returns edge surface area associated with the node. For 3-D volumetric solids, returns face area associated with the node.

Normals:	
NORMNX(*N1, N2, N3*)	X-direction cosine of the normal to the plane containing nodes *N1, N2, N3*
NORMNY(*N1, N2, N3*)	Y-direction cosine of the normal to the plane containing nodes *N1, N2, N3*
NORMNZ(*N1, N2, N3*)	Z-direction cosine of the normal to the plane containing nodes *N1, N2, N3*
NORMKX(*K1, K2, K3*)	X-direction cosine of the normal to the plane containing keypoints *K1, K2, K3*
NORMKY(*K1, K2, K3*)	Y-direction cosine of the normal to the plane containing keypoints *K1, K2, K3*
NORMKZ(*K1, K2, K3*)	Z-direction cosine of the normal to the plane containing keypoints *K1, K2, K3*

Connectivity:	
ENEXTN(*N,LOC*)	Element connected to node *N*. *LOC* is the position in the resulting list when many elements share the node. A zero is returned at the end of the list.
NELEM(*E, NPOS*)	Node number in position *NPOS* (1-20) of element *E*

Faces:	
ELADJ(*E, FACE*)	For 2-D planar solids and 3-D volumetric solids, element adjacent to a face (*FACE*) of element *E*. The face number is the same as the surface load key number. Only elements of the same dimensionality and shape are considered. A -1 is returned if more than one element is adjacent; A 0 is returned if there are no adjacent elements.
NDFACE(*E, FACE, LOC*)	Node in position *LOC* of a face number *FACE* of element *E*. The face number is the same as the surface load key number. *LOC* is the nodal position on the face (for an IJLK face, *LOC* = 1 is at node I, 2 is at node J, etc.).

Faces:	
NMFACE(*E*)	Face number of element *E* containing the selected nodes. The face number output is the surface load key. If multiple load keys occur on a face (such as for line and area elements), the lowest load key for that face is output.
ARFACE(*E*)	For 2-D planar solids and 3-D volumetric solids, returns the area of the face of element *E* containing the selected nodes. For axisymmetric elements, the area is the full (360°) area.

Degree of Freedom Results:	
UX(*N*)	UX structural displacement at node *N*
UY(*N*)	UY structural displacement at node *N*
UZ(*N*)	UZ structural displacement at node *N*
ROTX(*N*)	ROTX structural rotation at node *N*
ROTY(*N*)	ROTY structural rotation at node *N*
ROTZ(*N*)	ROTZ structural rotation at node *N*
TEMP(*N*)	Temperature at node *N*. For **SHELL131** and **SHELL132** elements with KEYOPT(3) = 0 or 1, use TBOT(*N*), TE2(*N*), TE3(*N*), . . ., TTOP(*N*) instead of TEMP(*N*).
PRES(*N*)	Pressure at node *N*
VX(*N*)	VX fluid velocity at node *N*
VY(*N*)	VY fluid velocity at node *N*
VZ(*N*)	VZ fluid velocity at node *N*
ENKE(*N*)	Turbulent kinetic energy (FLOTRAN) at node *N*
ENDS(*N*)	Turbulent energy dissipation (FLOTRAN) at node *N*
VOLT(*N*)	Electric potential at node *N*
MAG(*N*)	Magnetic scalar potential at node *N*
AX(*N*)	AX magnetic vector potential at node *N*
AY(*N*)	AY magnetic vector potential at node *N*
AZ(*N*)	AZ magnetic vector potential at node *N*

3.3.4. Listing Parameters

Once you have defined parameters, you can list them using the ***STATUS** command. If the ***STATUS** command is issued without arguments, it provides a list of all of the currently defined parameters. The following example shows the command and a typical listing.

```
*STATUS

PARAMETER STATUS-           (   5 PARAMETERS DEFINED)

NAME          VALUE          TYPE     DIMENSIONS
ABC       -24.0000000       SCALAR
HEIGHT     57.0000000       SCALAR
QR        2.070000000E+11   SCALAR
X_OR_Y    -24.0000000       SCALAR
CPARM       CASE1           CHARACTER
```

You can also access this information through either the **Utility Menu> List> Other> Parameters** or **Utility Menu> List> Status> Parameters> All Parameters** menu items.

Note — Any parameters beginning or ending in an underscore (_) are not shown by the ***STATUS** command.

You can check the status of individual parameters by providing these as arguments to the ***STATUS** command. The following example shows the status of the ABC parameter.

```
*STATUS,ABC

PARAMETER STATUS- abc       (   5 PARAMETERS DEFINED)

NAME         VALUE         TYPE     DIMENSIONS
ABC      -24.0000000      SCALAR
```

You can also check the status of specific parameters through the **Utility Menu> List> Other> Named Parameter** or **Utility Menu> List> Status> Parameters> Named Parameters** menu items.

Note — Although ANSYS allows a maximum of 5000 parameters, fewer than 5000 are available to the user due to GUI and ANSYS macro requirements. The number of parameters defined by the user interface (internal parameters) is listed by the ***STATUS** command. The command ***GET**,*par*,PARM,,MAX returns the total number of parameters defined.

3.4. Deleting Parameters

You can delete specific parameters in two ways:

- Issue the "=" command, leaving the right-hand side of the command blank. For example, to delete the QR parameter issue this command:

  ```
  QR=
  ```

- Issue the ***SET** command (**Utility Menu> Parameters> Scalar Parameters**), but don't specify a value for the parameter. For example, to delete the QR parameter via the ***SET** command issue the command as follows:

  ```
  *SET,QR,
  ```

Setting a numeric parameter equal to zero does not delete it. Similarly, setting a character parameter equal to empty single quotes (` `) or placing blanks within single quotes does not delete the parameter.

3.5. Using Character Parameters

Typically, character parameters are used to provide file names and extensions. The desired file name can be assigned to a character parameter, and that parameter can be used anywhere a file name is required. Similarly, a file extension can be assigned to a character parameter and used where appropriate (typically the *Ext* command argument). In batch mode, this allows you to easily change file names for multiple runs by simply changing the initial alphanumeric "value" of the character parameter in your input file.

Note — Remember that character parameters are limited to a total of eight characters.

The following is a list of general uses for character parameters.

- As arguments to any applicable command field (that is, where alphanumeric input is expected)
- As macro name arguments for the ***USE** command (**Utility Menu> Macro> Execute Data Block**)

  ```
  NAME='MACRO'   ! MACRO is the name of a macro file
  *USE,NAME      ! Calls MACRO
  ```

- As arguments to macro calls for ***USE** and for the "unknown command" macro. Any of the following macro calls are allowed:

```
ABC='SX'
*USE,NAME,ABC
```

or

```
*USE,NAME,'SX'

DEF='SY'
NEWMACRO,DEF     ! Calls existing macro file NEWMACRO.MAC
```

or

```
NEWMACRO,'SY'
```

3.6. Substitution of Numeric Parametric Values

Whenever you use a parameter name in a numeric command field, its value is automatically substituted. If no value has been assigned to the parameter (that is, if the parameter has not been defined), a near-zero value (2^{-100}) will be substituted, usually without warning.

Note — Defining the parameter after it is used in a command does not "update" the command in *most* cases. (Exceptions are the commands **/TITLE**, **/STITLE**, ***ABBR**, and **/TLABEL**. See Section 3.6.2.1: Forced Substitution for more information.) For example:

```
Y=0
X=2.7
N,1,X,Y     ! Node 1 at (2.7,0)
Y=3.5       ! Redefining parameter Y now does not update node 1
```

3.6.1. Preventing Substitution

You can prevent parameter substitution by enclosing the parameter name with single quotes ('), for example, 'XYZ'. The literal string is then used; therefore, this feature is valid only in *non-numerical* fields.

Conversely, you can force parameter substitution in titles, subtitles, and filenames by enclosing the parameter name with percent signs (%). For example,

```
/TITLE, TEMPERATURE CONTOURS AT TIME=%TM%
```

specifies a title in which the numerical value of parameter TM is substituted. Note that the parameter is substituted at the time the title is used.

3.6.2. Substitution of Character Parametric Values

Use of a character parameter in an alphanumeric command field generally results in automatic substitution of its value. Forced substitution and character parameter restrictions are explained below.

3.6.2.1. Forced Substitution

As with numerical parameters, you can force the substitution of a character parameter value in certain cases where substitution would not occur otherwise. This is done by enclosing the character parameter name with percent signs (%). Forced substitution of character parameters is valid for the following commands:

- **/TITLE** command (*Title* field). Specifies titles for various printed output.
- **/STITLE** command (*Title* field). Specifies subtitles, similar to**/TITLE**. (You cannot access the **/STITLE** command directly in the GUI.)

- **/TLABEL** command (*Text* field). Specifies text string for annotation.
- ***ABBR** command (*Abbr* field). Defines an abbreviation.

Forced substitution is also valid in the following types of fields:

- Any filename or extension command argument. These arguments apply to commands such as **/FILNAME**, **RESUME**, **/INPUT**, **/OUTPUT**, and **FILE**. (Direct parameter substitution is also valid in these fields.)
- Any 32 character field: A typical example is the name of macros. (Direct substitution is not valid for these fields.)
- As a command name in any command name field. Also as an "unknown command" macro name in field 1. For example:

```
R='RESUME'
%R%,MODEL,DB
```

The following example of the command input method shows forced substitution for a subtitle definition and for a directory name.

```
A='TEST'
B='.RST'
C='/ANSYS'
D='/MODELS/'
/STITLE,,RESULTS FROM FILE %C%%D%%A%%B%

 SUBTITLE  1 =
 RESULTS FROM FILE /ANSYS/MODELS/TEST.RST

/POST1
FILE,A,RST,%C%%D%      ! Read results from /ANSYS/MODELS/TEST.RST
```

3.6.2.2. Other Places Where Character Parameters Are Valid

In addition to the more general applications already discussed, there are some specific instances where character parameters are allowed for added convenience. The commands which are affected and details of usage are outlined below.

***ASK**
This command may prompt you for an alphanumeric string (up to eight characters enclosed in single quotes) which is assigned to a character scalar parameter. (You cannot access the ***ASK** command directly in the GUI.)

***CFWRITE**
This command writes ANSYS commands to the file opened by ***CFOPEN**. It can be used to write a character parameter assignment to that file. For example, ***CFWRITE**,B = 'FILE' is valid. (You cannot access the ***CFWRITE** and ***CFOPEN** commands directly in the GUI.)

***IF** and ***ELSEIF**
Character parameters may be used for the *VAL1* and *VAL2* arguments of these commands. For the *Oper* argument, only labels EQ (equal) and NE (not equal) are valid when using character parameters. (You cannot access the ***IF** and ***ELSEIF** commands directly in the GUI.) Example:

```
CPARM='NO'
*IF,CPARM,NE,'YES',THEN
```

***MSG**
Character parameters are allowed as input for the *VAL1* through *VAL8* arguments. The data descriptor %C is used to indicate alphanumeric character data on the format line (which must follow the ***MSG** command).

The %C corresponds to the FORTRAN descriptor A8. (You cannot access the ***MSG** command directly in the GUI.)

PARSAV and **PARRES**
These commands will save character parameters to a file (**PARSAV** command or menu path **Utility Menu> Parameters> Save Parameters**) and resume character parameters from a file (**PARRES** or **Utility Menu> Parameters> Restore Parameters**).

***VREAD**
This command (**Utility Menu> Parameters> Array Parameters> Read from File)** can be used to read alphanumeric character data from a file and produce a character array parameter. The FORTRAN character descriptor (A) may be used in the format line which must follow the ***VREAD** command.

***VWRITE**
This command (menu path **Utility Menu> Parameters> Array Parameters> Write to File**) can be used to write character parameter data to a file in a formatted sequence. The FORTRAN character descriptor (A) may be used in the format line which must follow the ***VWRITE** command.

3.6.2.3. Character Parameter Restrictions

Although character parameters have much of the same functionality as numerical parameters, there are several instances where character parameters are not valid.

- Character parameter substitution is not allowed for the *Par* argument of the ***SET**, ***GET**, ***DIM**, and ***STATUS** commands.
- Interactive editing of array parameters (***VEDIT** command) is not available for character array parameters.
- Vector operation commands, such as ***VOPER**, ***VSCFUN**, ***VFUN**, ***VFILL**, ***VGET**, and ***VITRP**, do not work with character array parameters.
- When operating on character parameters, the specification commands ***VMASK** and ***VLEN** are applicable only to the ***VWRITE** and ***VREAD** commands.
- Character parameters are not valid in parametric expressions which use addition, subtraction, multiplication, etc.

3.7. Dynamic Substitution of Numeric or Character Parameters

Dynamic substitution of parameters will occur for the following commands: **/TITLE**, **/STITLE**, ***ABBR**, **/AN3D**, and **/TLABEL**. Dynamic substitution allows the revised value of a parameter to be used, even if the command which uses the parameter value has not been reissued.

Example:

```
XYZ='CASE 1'
/TITLE,This is %XYZ%
APLOT
```

The title "This is CASE 1" will appear on the area plot.

You can then change the value of XYZ and the new title will appear on subsequent plots, even though you did not reissue **/TITLE**.

```
XYZ='CASE 2'
```

The title "This is CASE 2" will appear on subsequent plots.

3.8. Parametric Expressions

Parametric expressions involve operations among parameters and numbers such as addition, subtraction, multiplication, and division. For example:

```
X=A+B
P=(R2+R1)/2
D=-B+(E**2)-(4*A*C)     ! Evaluates to D = -B + E2 - 4AC
XYZ=(A<B)+Y**2          ! Evaluates to XYZ = A + Y2 if A is less than B;
                        ! otherwise to XYZ = B + Y2
INC=A1+(31.4/9)
M=((X2-X1)**2-(Y2-Y1)**2)/2
```

The following is a complete list of APDL operators:

Operator	Operation
+	Addition
_	Subtraction
*	Multiplication
/	Division
**	Exponentiation
<	Less-Than Comparison
>	Greater-Than Comparison

You can also use parentheses for clarity and for "nesting" of operations, as shown above. The order in which the ANSYS program evaluates an expression is as follows:

1. Operations in parentheses (innermost first)
2. Exponentiation (in order, from right to left)
3. Multiplication and division (in order, from left to right)
4. Unary association (such as +A or -A)
5. Addition and subtraction (in order, from left to right)
6. Logical evaluation (in order, from left to right)

Thus an expression such as Y2=A+B**C/D*E will be evaluated in this order: B**C first, /D second, *E third, and +A last. For clarity, you should use parentheses in expressions such as these. Parentheses can be nested up to four levels deep, and up to nine operations can be performed within each set of parentheses. As a general rule, avoid using blank spaces between operators in expressions. In particular, never include a blank space before the * character because the rest of the input line (beginning with the *) will be interpreted as a comment and therefore will be ignored. (Do not use this convention as a comment; use an exclamation point (!) for this purpose.)

3.9. Parametric Functions

A parametric function is a programmed sequence of mathematical operations which returns a single value, such as SIN(X), SQRT(B), and LOG(13.2). The following table provides a complete list of functions currently available in ANSYS.

ABS(x)	Absolute value of x.
SIGN(x,y)	Absolute value of x with sign of y. y=0 results in positive sign.
EXP(x)	Exponential of x (e^x).

LOG(x)	Natural log of x (ln (x)).
LOG10(x)	Common log of x ($\log_{10}$(x)).
SQRT(x)	Square root of x.
NINT(x)	Nearest integer to x.
MOD(x,y)	Remainder of x/y. y=0 returns zero (0).
RAND(x,y)	Random number (uniform distribution) in the range x to y (x = lower bound, y = upper bound).
GDIS(x,y)	Random sample of a Gaussian (normal) distribution with mean x and standard deviation y.
SIN(x), COS(x), TAN(x)	Sine, Cosine, and Tangent of x. x is in radians by default, but can be changed to degrees with ***AFUN**.
SINH(x), COSH(x), TANH(x)	Hyperbolic sine, Hyperbolic cosine, and Hyperbolic tangent of x.
ASIN(x), ACOS(x), ATAN(x)	Arcsine, Arccosine, and Arctangent of x. x must be between -1.0 and +1.0 for ASIN and ACOS. Output is in radians by default, but can be changed to degrees with ***AFUN**. Range of output is -pi/2 to +pi/2 for ASIN and ATAN, and 0 to pi for ACOS.
ATAN2(y,x)	Arctangent of y/x with the sign of each component considered. Output is in radians by default, but can be changed to degrees with ***AFUN**. Range of output is -pi to +pi.
VALCHR (*CPARM*	Numerical value of *CPARM* (if *CPARM* is non-numeric, returns 0.0).
CHRVAL (*PARM*)	Character value of numerical parameter *PARM*. Number of decimal places depends on magnitude.
UPCASE *CPARM*	Upper case equivalent of *CPARM*.
LWCASE *(CPARM)*	Lower case equivalent of *CPARM*.

The following are examples of parametric functions:

```
PI=ACOS(-1)             ! PI = arc cosine of -1, PI calculated to machine accuracy
Z3=COS(2*THETA)-Z1**2
R2=SQRT(ABS(R1-3))
X=RAND(-24,R2)          ! X = random number between -24 and R2

*AFUN,DEG               ! Units for angular functions are degrees
THETA=ATAN(SQRT(3))     ! THETA evaluates to 60 degrees
PHI=ATAN2(-SQRT(3),-1)  ! PHI evaluates to -120 degrees
*AFUN,RAD               ! Units for angular functions reset to radians

X249=NX(249)            ! X-coordinate of node 249
SLOPE=(KY(2)-KY(1))/(KX(2)-KX(1))
                        ! Slope of line joining keypoints 1 and 2

CHNUM=CHRVAL(X)         ! CHNUM = character value of X
UPPER=UPCASE(LABEL)     ! UPPER = uppercase character value of parameter LABEL
```

3.10. Saving, Resuming, and Writing Parameters

If you must use currently defined parameters in another ANSYS session, you can write them to a file and then read (resume) that file. When you read the file, you can either completely replace currently defined parameters or add to them (replacing those that already exist).

To write parameters to a file, use the **PARSAV** command (**Utility Menu> Parameters> Save Parameters**).

The parameters file is an ASCII file consisting largely of APDL ***SET** commands used to define the various parameters. The following example shows the format of this file.

```
/NOPR
*SET,A       ,  10.00000000000
*SET,B       ,  254.3948750000
*SET,C       ,'string  '
*SET,_RETURN , 0.00000000000000E+00
*SET,_STATUS ,  1.000000000000
*SET,_ZX     ,'        '
/GO
```

To read parameters from a file use the **PARRES** command (**Utility Menu> Parameters> Restore Parameters**)

If you wish, you can write up to ten parameters or array parameters using FORTRAN real formats to a file. You can use this feature to write your own output file for use in other programs, reports, etc. To do this, use the ***VWRITE** command (**Utility Menu> Parameters> Array Parameters> Write to File**). The ***VWRITE** command is discussed in Section 3.11.7: Operations Among Array Parameters.

3.11. Array Parameters

In addition to scalar (single valued) parameters, you can define array (multiple valued) parameters. ANSYS arrays can be

- 1-D (a single column)
- 2-D (rows and columns)
- 3-D (rows, columns, and planes)
- 4-D (rows, columns, planes, and books)
- 5-D (rows, columns, planes, books, and shelves)

ANSYS provides three types of arrays:

ARRAY
: This type is similar to FORTRAN 77 arrays and is the default array type when dimensioning arrays. As with FORTRAN arrays, the indices for rows, columns, and planes are sequential integer numbers beginning with one. Array elements can be either integers or real numbers.

CHAR
: This is a character array, with each element consisting of an alphanumeric value not exceeding eight characters. The indices for rows, columns, and planes are sequential integer numbers beginning with one.

TABLE
: This is a special type of numeric array which allows ANSYS to calculate (through linear interpolation) values between these array elements explicitly defined in the array. Moreover, you can define the array indices for each row, column, and plane and these indices are real (not integer) numbers. Array elements can be either integers or real numbers. As we'll see in the later discussion on TABLE arrays, this capability provides a powerful method for describing mathematical functions.

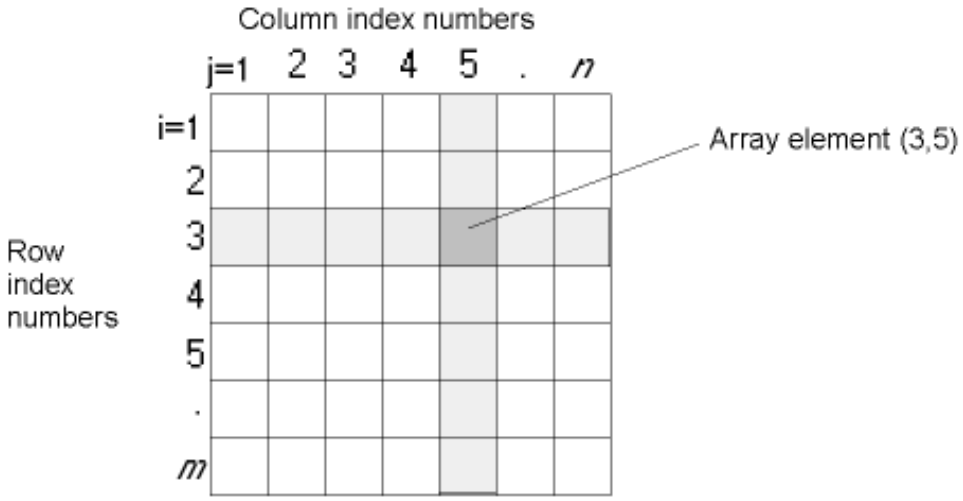
STRING
: You can use the ***DIM**, STRING capability to enter character strings into your arrays. Index numbers for columns and planes are sequential values beginning with 1. Row indices are determined by the character position in the string. See the ***DIM** command for more information.

All three types of arrays cannot exceed 2*31-1 bytes. For a double precision array, each data item is 8 bytes, so the limit on number of entries is (2**31-1)/8.

3.11.1. Array Parameter Basics

Consider a 2-D array (either ARRAY or CHAR) as shown below. It is *m* rows long and *n* columns wide; that is, its dimensions are *m* times *n*. Each row is identified by a row index number *i*, which varies from 1 to *m*, and each column is identified by a column index number *j*, which varies from 1 to *n*. The quantities that make up the array are array elements. Each array element is identified as *(i,j)*, where *i* is its row index number and *j* is its column index number.

Figure 3.1 A Graphical Representation of a 2-D Array



We can extend these definitions to a 3-D array parameter, which may be *m* rows long, *n* columns wide, and *p* planes deep. The plane index number is *k*, which varies from 1 to *p*. Each array element is identified as (*i*, *j*, *k*,). The following figure shows a 3-D array.

Figure 3.2 A Graphical Representation of a 3-D Array

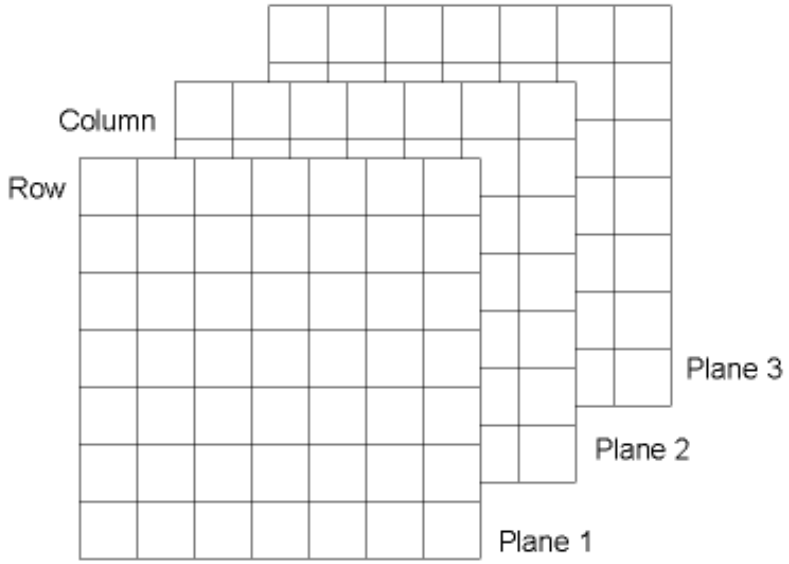
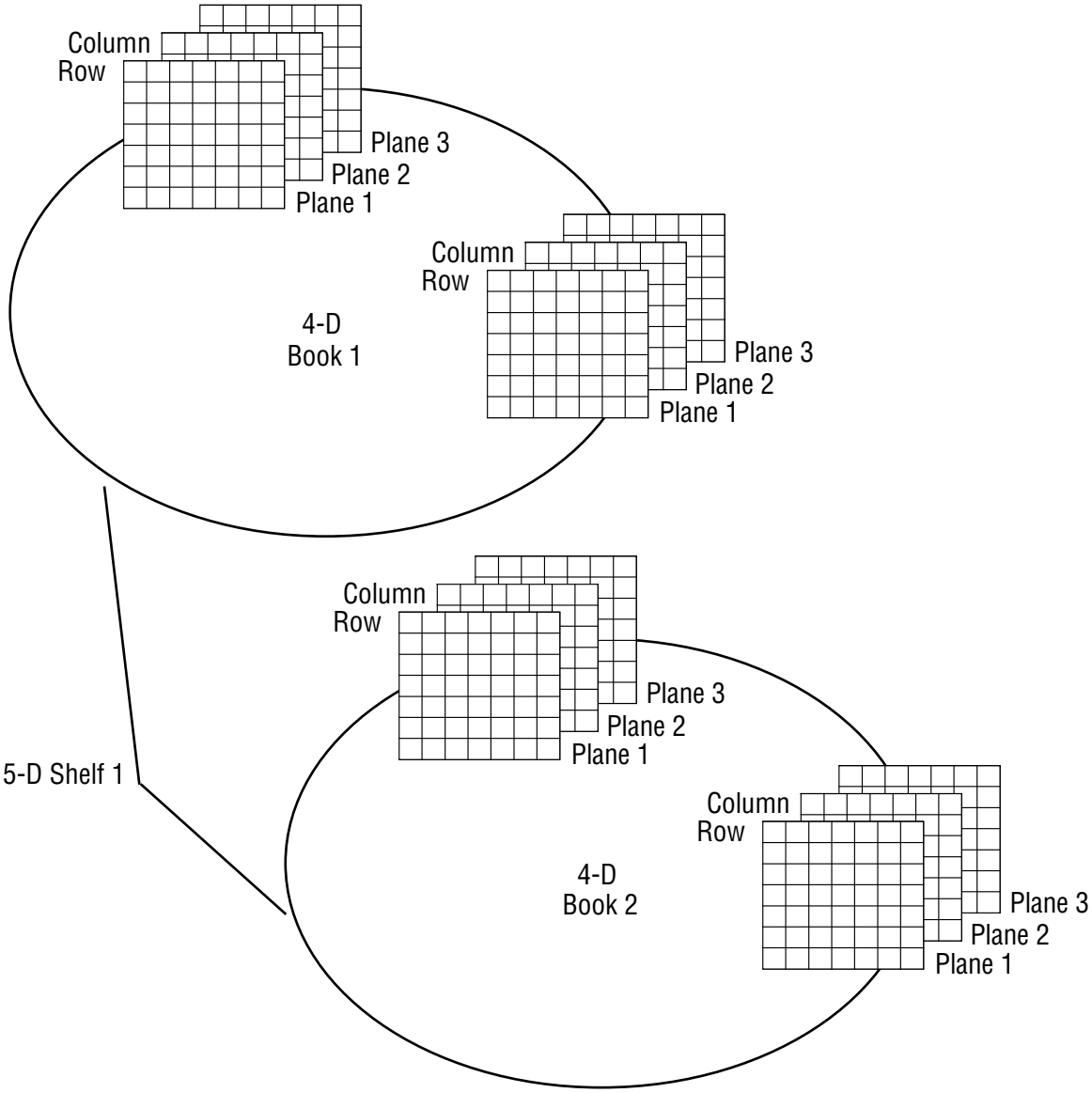


Figure 3.3 A Graphical Representation of a 5-D Array



3.11.2. Array Parameter Examples

Type ARRAY parameters consist of discrete numbers that are simply arranged in a tabular fashion for convenience. Consider the following examples.

$$\text{NTEMP} = \begin{bmatrix} -47.6 \\ -5.2 \\ 25.0 \\ 86.5 \\ 107.9 \\ 168.7 \\ 225.0 \end{bmatrix} \qquad \text{EVOLUM} = \begin{bmatrix} 0.025 \\ 0.01 \\ 0.265 \\ 1.00 \\ 0.832 \\ 0.52 \\ 1.032 \\ 0.002 \\ 0.697 \\ 0.01 \end{bmatrix}$$

$$\text{COMPSTRS} = \begin{bmatrix} 12152 & 814 & -386 & 202 & -82 & -1108 \\ 14848 & 1057 & -704 & 117 & -101 & -555 \\ 15490 & 1033 & -713 & 15 & -76 & 235 \\ 13899 & 786 & -348 & -103 & -45 & 848 \\ 10813 & 420 & -66 & -211 & -17 & 1065 \\ 7151 & 109 & 111 & -272 & 11 & 1052 \end{bmatrix}$$

The parameter NTEMP could be an array of temperatures at selected nodes; NTEMP(1) = -47.6 could be the temperature at node 27, NTEMP(2) = -5.2 could be the temperature at node 43, and so on. Similarly, EVOLUM could be an array of element volumes, and COMPSTRS could be an array of nodal component stresses, with each column representing a particular direction (X, Y, Z, XY, YZ, XZ, for example).

A type CHAR array parameter is structured similarly to an ARRAY parameter, with the tabular values being alpha-numeric character strings (up to eight characters). Two examples of character array parameters are:

$$\text{FILNAM} = \begin{bmatrix} \text{JOB1} \\ \text{JOB2} \\ \text{JOB3} \\ \text{JOB4} \\ \text{JOB5} \end{bmatrix} \qquad \text{EXTENS} = \begin{bmatrix} \text{LOG} \\ \text{ERR} \\ \text{DB} \\ \text{LIB} \\ \text{MAC} \end{bmatrix}$$

3.11.3. TABLE Type Array Parameters

A type TABLE array parameter consists of numbers (alphanumeric values are not valid) arranged in a tabular fashion, much like the ARRAY type. However, there are three important differences

- ANSYS can calculate (through linear interpolation) any values that fall between the explicitly declared array element values.
- A table array contains a 0 row and 0 column used for data-access index values, and unlike standard arrays, these index values can be real numbers. The only restriction is that the index values must be numerically

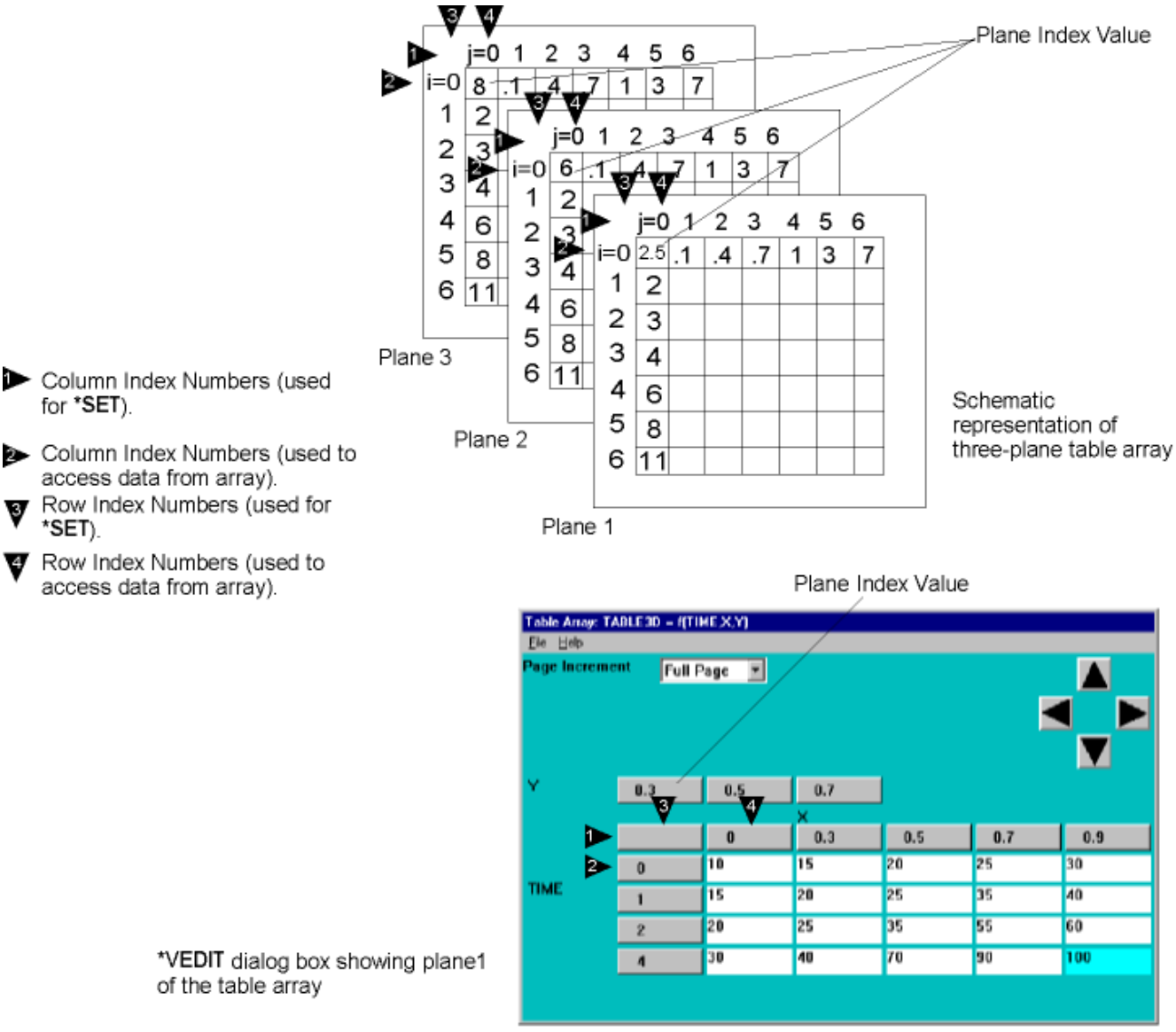
increasing (never decreasing) numbers. You must explicitly declare a data access index value for each row and column; otherwise the default value assigned is the "tiny number" (7.888609052E-31).

You can more conveniently define the index starting point and index values via the ***TAXIS** command.

- A plane index value resides in the 0,0 location for each plane.

The following figure shows a TABLE array with data-access index values. Note that the indexes are specified as the "0" row and column values.

Figure 3.4 A Graphical Representation of a Table Array



As shown in the above example, when configuring a table array you must set

- The plane index value as the 0,0 element value for *each* plane.
- The data-access column index values in the elements in the *0 row in plane 1*. These values are used only when accessing data from the array. When setting the array element values, you use the traditional row and column index numbers.

- The data-access row index values in the elements in the *0 column in plane 1*. Again, these values are used only when accessing data from the array. When setting the array element values, you use the traditional row and column index numbers.

3.11.4. Defining and Listing Array Parameters

To define an array parameter, you must first declare its type and dimensions using the ***DIM** command (**Utility Menu> Parameters> Array Parameters> Define/Edit**).

This following examples illustrate the ***DIM** command used to dimension various types of arrays:

```
*DIM,AA,,4              ! Type ARRAY is default, dimension 4[x1x1]
*DIM,XYZ,ARRAY,12       ! Type ARRAY array, dimension 12[x1x1]
*DIM,FORCE,TABLE,5      ! Type TABLE array, dimension 5[x1x1]
*DIM,T2,,4,3            ! Dimensions are 4x3[x1]
*DIM,CPARR1,CHAR,5      ! Type CHAR array, dimension 5[x1x1]
```

Note — Array elements for ARRAY and TABLE are initialized to 0 (except for the 0 row and column for TABLE, which is initialized to the tiny value). Array elements for CHAR are initialized to a blank value.

The next example shows how to fill a 5-D array with data. Use 1-D tables to load a 5-D table. Use the ***TAXIS** to define the table index values. See the full example at Section 2.6.14.6: Example Analysis Using 5-D Table Array.

```
*dim,xval,array,X1
*dim,yval,array,Y1
yval(1)=0,20
*dim,zval,array,10
zval(1)=10,20,30,40,50,60,70,80,90,100
*dim,tval,array,5
tval(1)=1,.90,.80,.70,.60
*dim,tevl,array,5
tevl(1)=1,1.20,1.30,1.60,1.80

*dim,ccc,tab5,X1,Y1,Z1,D4,D5,X,Y,Z,TIME,TEMP
*taxis,ccc(1,1,1,1,1),1,0,wid                    !!! X-Dim
*taxis,ccc(1,1,1,1,1),2,0,hth                    !!! Y-Dim
*taxis,ccc(1,1,1,1,1),3,1,2,3,4,5,6,7,8,9,10     !!! Z-Dim
*taxis,ccc(1,1,1,1,1),4,0,10,20,30,40             !!! Time
*taxis,ccc(1,1,1,1,1),5,0,50,100,150,200         !!! Temp
*do,ii,1,2
   *do,jj,1,2
      *do,kk,1,10
         *do,ll,1,5
            *do,mm,1,5
               ccc(ii,jj,kk,ll,mm)=(xval(ii)+yval(jj)+zval(kk))*tval(ll)*tevl(mm)
            *enddo
         *enddo
      *enddo
   *enddo
*enddo
```

3.11.5. Specifying Array Element Values

You can specify array element values by

- Setting individual array element values through the ***SET** command or "=" shortcut.
- Filling individual vectors (columns) in the array with either specified or calculated values (the ***VFILL** command, for example).
- Interactively specifying values for the elements through the ***VEDIT** dialog box.
- Reading the values from an ASCII file (***VREAD** or ***TREAD** commands).

Note — You cannot create or edit 4- or 5-D arrays interactively. ***VEDIT**, ***VREAD**, and ***TREAD** are not applicable to 4- or 5-D arrays.

3.11.5.1. Specifying Individual Array Values

You can use either the ***SET** command or the "=" shortcut. Usage is the same as for scalar parameters, except that you now define a column of data (up to ten array element values per "=" command). For example, to define the parameter XYZ dimensioned above as a 12x1 array you will need two "=" commands. In the following example the first command defines the first eight array elements and the second command defines the next four array elements:

```
XYZ(1)=59.5,42.494,-9.01,-8.98,-8.98,9.01,-30.6,51
XYZ(9)=-51.9,14.88,10.8,-10.8
```

$$XYZ = \begin{bmatrix} 59.5 \\ 42.494 \\ -9.01 \\ -8.98 \\ -8.98 \\ 9.01 \\ -30.6 \\ 51 \\ -51.9 \\ 14.88 \\ 10.8 \\ -10.8 \end{bmatrix}$$

Notice that the starting location of the array element is indicated by the row index number of the parameter (1 in the first command, 9 in the second command).

The following example shows how to define the element values for the 4x3 array parameter T2, dimensioned earlier in the ***DIM** examples:

```
T2(1,1)=.6,2,-1.8,4          ! defines (1,1),(2,1),(3,1),(4,1)
T2(1,2)=7,5,9.1,62.5         ! defines (1,2),(2,2),(3,2),(4,2)
T2(1,3)=2E-4,-3.5,22,.01     ! defines (1,3),(2,3),(3,3),(4,3)
```

$$T2 = \begin{bmatrix} 0.6 & 7.0 & 0.0002 \\ 2.0 & 5.0 & -3.5 \\ -1.8 & 9.1 & 22.0 \\ 4.0 & 62.5 & 0.01 \end{bmatrix}$$

The following example defines element values for the TABLE array parameter FORCE discussed earlier.

```
FORCE(1)=0,560,560,238.5,0
FORCE(1,0)=1E-6,.8,7.2,8.5,9.3
```

The first "=" command defines the five array elements of the TABLE array FORCE. The second and third "=" commands redefine the index numbers in the j=0 and i=0 row.

$$\text{FORCE} = \begin{array}{c} \\ 1E-6 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 0 \\ \left[\begin{array}{c} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{array}\right. \end{array}$$

Character array parameters can also be defined using the "**=**" command. Assigned values can be up to eight characters each and must be enclosed in single quotes. For example:

```
*DIM,RESULT,CHAR,3       !Character array parameter with dimensions (3,1,1)
RESULT(1)='SX','SY','SZ'  !Assigns values to parameter RESULT
```

Notice that, as when defining a numerical array parameter, the starting location of the array element must be specified (in this case, the row index number 1 is indicated).

Note — CHAR cannot be used as a character parameter name because it will create a conflict with the CHAR label on the ***DIM** command. ANSYS will substitute the character string value assigned to parameter CHAR when CHAR is input on the third field of the ***DIM** command (Type field).

3.11.5.2. Filling Array Vectors

You can use the ***VFILL** command (**Utility Menu> Parameters> Array Parameters> Fill**) to "fill" an ARRAY or TABLE vector (column).

See the ***VFILL** command reference information in the *ANSYS Commands Reference* for more detail about the command syntax. The following example illustrates the capabilities of the ***VFILL** command.

```
*DIM,DTAB,ARRAY,4,3                    ! dimension 4 x 3 numeric array
*VFILL,DTAB(1,1),DATA,-3,8,-12,57      ! four data values loaded into vector 1
*VFILL,DTAB(1,2),RAMP,2.54,2.54        ! fill vector 2 with values starting at
                                       ! 2.54 and incrementing by 2.54
*VFILL,DTAB(1,3),RAND,1.5,10           ! fill vector 3 with random numbers between
                                       ! 1.5 and 10.  Results will vary due to
                                       ! random number generation.
```

$$\text{DTAB} = \begin{bmatrix} -3 & 2.54 & 2.799901284 \\ 8 & 5.08 & 6.11292418 \\ -12 & 7.62 & 6.70205516 \\ 57 & 10.16 & 4.11487684 \end{bmatrix}$$

3.11.5.3. Interactively Editing Arrays

The ***VEDIT** command (**Utility Menu> Parameters> Array Parameters> Define/Edit**), which is available only in interactive mode, launches a data entry dialog box you can use to edit an ARRAY or TABLE (not CHAR) array. The dialog box provides a number of convenient features:

- A spreadsheet-style editor for array element values.
- Navigational controls for scrolling through large arrays.
- An initialize function to set any row or column to a specified value (ARRAY type only).

- Delete, copy, and insert functions for moving rows or columns of data (ARRAY type only).

Complete instructions for using the dialog box are available from the box's Help button.

Note — You cannot edit a 4- or 5-D ARRAY or TABLE interactively.

Figure 3.5 An Example *VEDIT Dialog Box for an ARRAY

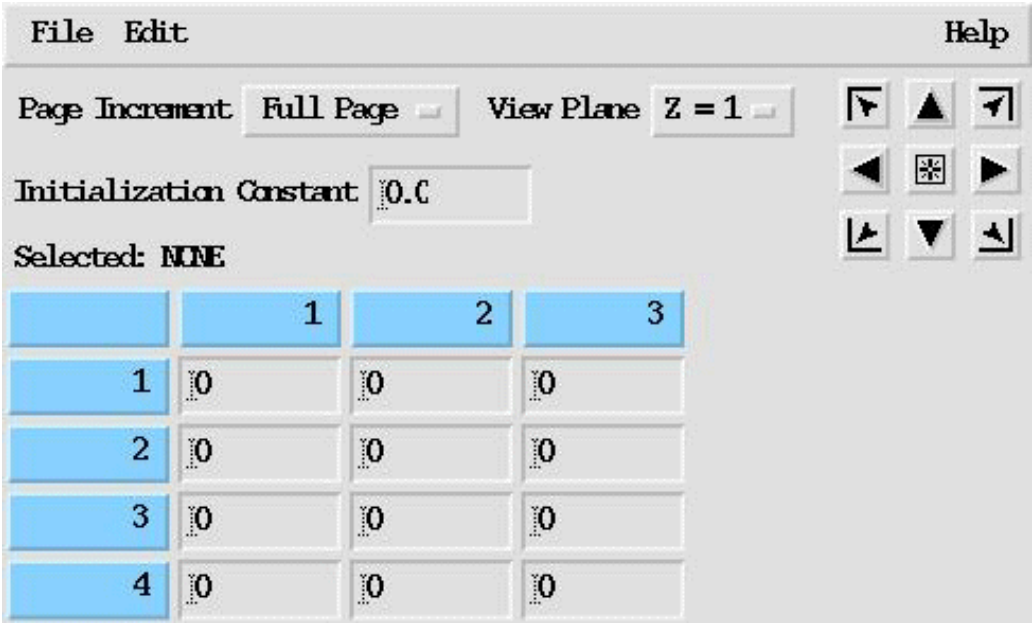
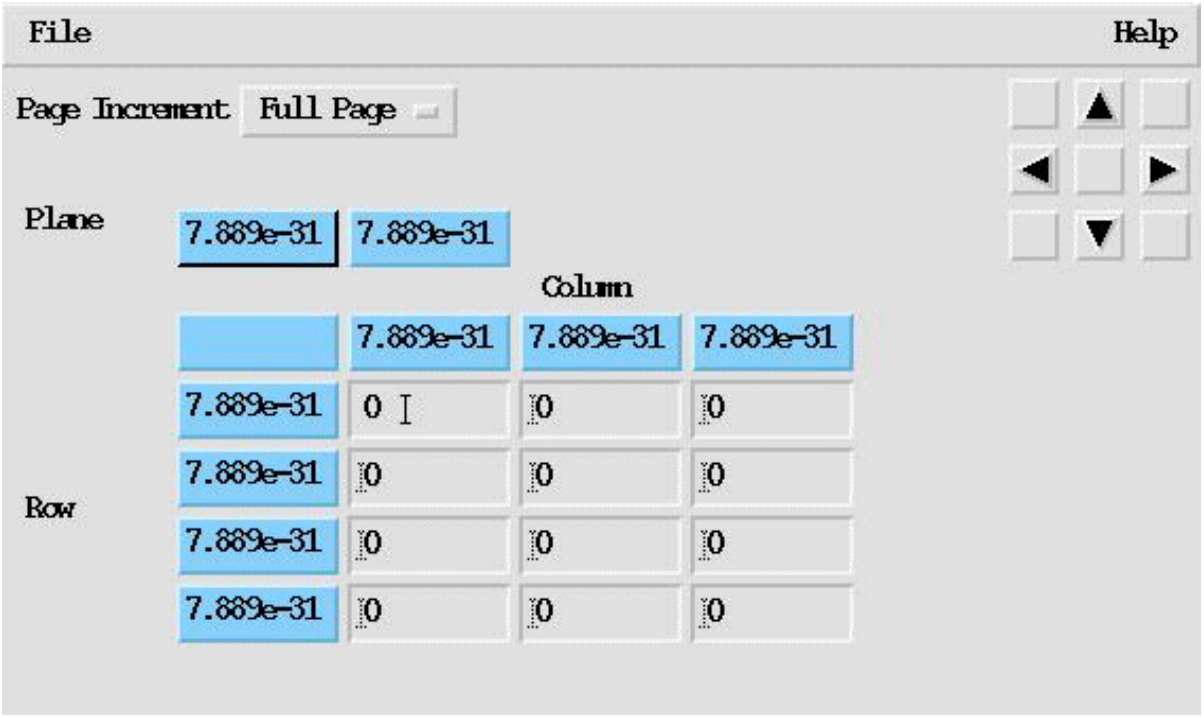


Figure 3.6 An Example *VEDIT Dialog Box for a TABLE



3.11.5.4. Filling an Array From a Data File Using *VREAD

You can fill an array from a data file using the ***VREAD** command (**Utility Menu> Parameters> Array Parameters> Read from File**). The command reads information from an ASCII data file and begins writing it into the array, starting with the index location that you specify. You can control the format of the information read from the file through data descriptors. The data descriptors must be enclosed in parenthesis and placed on the line following the ***VREAD** command. See Section 3.11.7.1: Vector Operations for more information about data descriptors. The data descriptors control the number of fields to be read from each record, the width of the data fields, and the position of the decimal point in the field.

For example, given the following data file named **dataval**:

```
1.5     7.8  12.3
15.6  -45.6  42.5
```

and an array called EXAMPLE that has been dimensioned as 2 x 3, the following commands (provided as either a part or a macro or input listing)

```
*DIM,EXAMPLE,,2,3
*VREAD,EXAMPLE(1,1),dataval,,,JIK,3,2
(3F6.1)
```

result in

```
                                  1.5     7.8  12.3
EXAMPLE =                         15.6  -45.6  42.5
```

The ***VREAD** command cannot be issued directly from the command input window. However, the **Utility Menu> Parameters> Array Parameters> Read from File** dialog box offers a way to specify the data descriptors and issue the command in interactive mode.

Note — You cannot fill a 4- or 5-D array using ***VREAD**.

3.11.5.5. Filling a TABLE Array From a Data File Using *TREAD

Once configured, you have two options for specifying values for the TABLE array elements: you can add values as you would for any other type of array, or you can read in a table of data from an external file.

To read in a table of data from an external file, you still define the TABLE array first, specifying the number of rows, columns, and planes, and the labels for each. You can then read an ASCII file containing the table of data using the ***TREAD** command (**Utility Menu> Parameters> Array Parameters> Read from File**). At this time, you also specify the number of lines to skip (NSKIP) between the top of the file and the first line of the table.

When reading data from an external file, remember:

- The file containing the table of data can be created in a text editor or an external application (such as Microsoft Excel), but it must be in ASCII form, tab-delimited, to be read into ANSYS.
- You must first define the array in ANSYS, remembering to allow for the index values (0,0).
- The values are read straight across the rows until all columns on each row of the array are filled; ANSYS then wraps from one row to the next and begins to fill those columns, and so on. Be sure that the dimensions of the array you defined are correct. If you mistakenly define fewer columns in the ANSYS array than required, ANSYS will start filling in the next row of the array using the values remaining in the first row of the data table being read. Similarly, if you define more columns in the ANSYS array than required, ANSYS will fill all columns of the array using values from the next row of the data table being read, and only then wrap and begin filling the next row.

You can create 1-D, 2-D, and 3-D tables by reading data from an external file. Examples of how you create each of these follows.

Note — You cannot fill a 4- or 5-D TABLE using ***TREAD**.

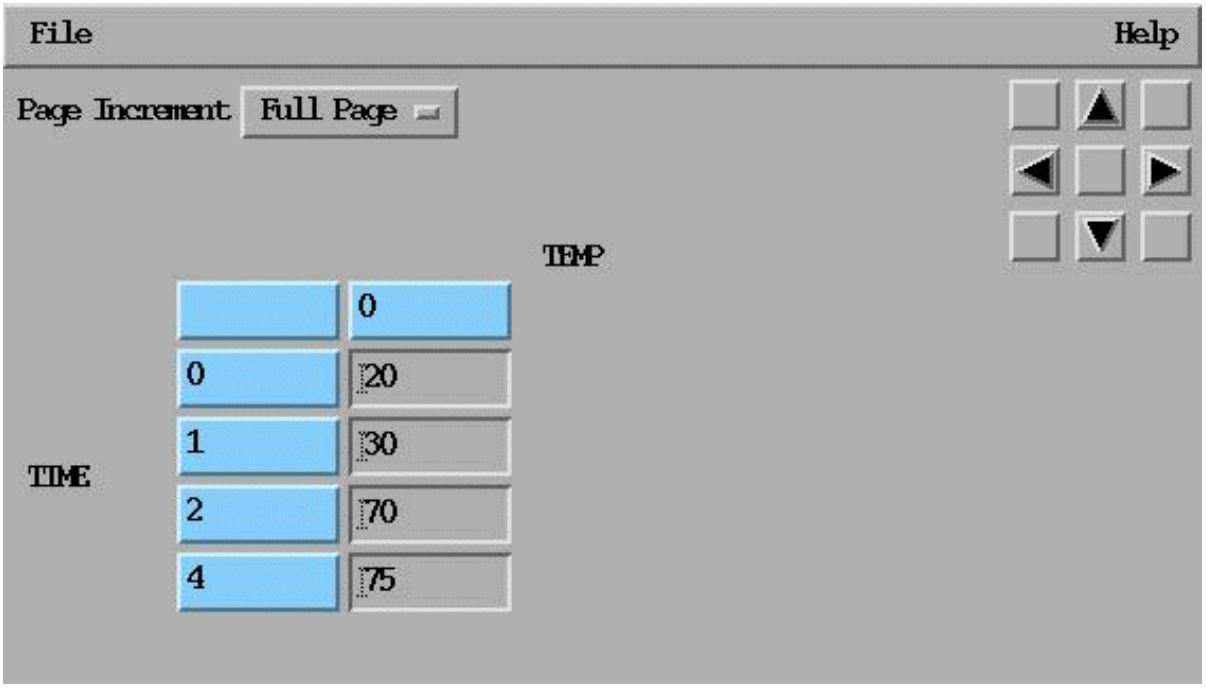
Example 1: 1-D Table

First, create the 1-D table using the application of your choice (such as a spreadsheet application, a text editor, etc.) and then save the file as a text file in tab-delimited format. In this example, the table is named "Tdata" and contains data for time vs. temperature. In its ASCII form, the table would look like this:

Time Temperature Table	
Time	Temp
0	20
1	30
2	70
4	75

In ANSYS, you define a TABLE parameter "Tt" using the ***DIM** command (**Utility Menu> Parameters> Array Parameters> Define/Edit**). Specify 4 rows and 1 column, row label of Time, and column label of Temp. Note that the data table you created has four rows and one column of data, plus the row and column index values (the first column - TIME - is the row index values) Then read in the file as described earlier, specifying 2 skipped lines. The TABLE array in ANSYS would look like this:

Figure 3.7 A Sample 1-D TABLE Array Dialog Box



This same example, done via command input, would look like the following:

```
*DIM,Tt,table,4,1,1,TIME,TEMP
*TREAD,Tt,tdata,txt,,2
```

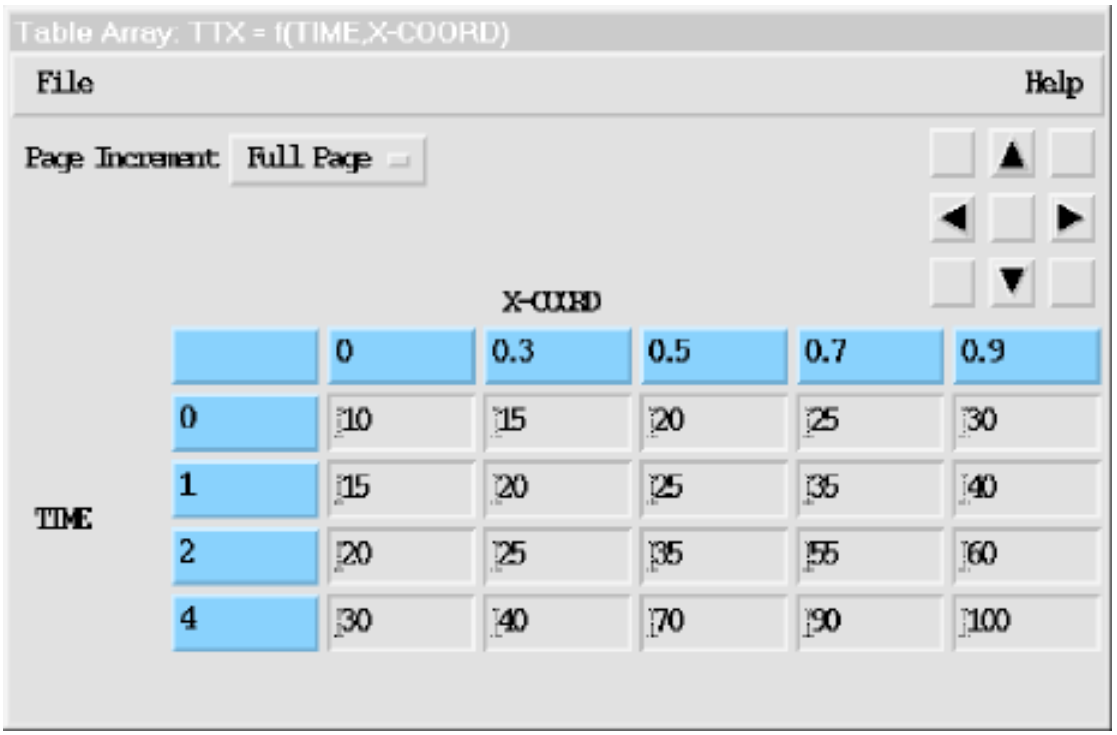
Example 2: 2-D Table

For this example, create (in a spreadsheet application, a text editor, etc.) a 2-D table named "T2data" containing temperature data as a function of time and x-coordinate and read it into a TABLE array parameter called "Ttx." The table, in its ASCII form, would look like this:

Temp (time-X-coord) Table					
Time	X-Coordinate				
0	0	.3	.5	.7	.9
0	10	15	20	25	30
1	15	20	25	35	40
2	20	25	35	55	60
4	30	40	70	90	100

In ANSYS, you define a TABLE parameter "Ttx" using the ***DIM** command (**Utility Menu> Parameters> Array Parameters> Define/Edit**). Specify 4 rows, 5 columns, 1 plane, row label of TIME, and column label of X-COORD. Note that the data table you created has four rows and five columns of data, plus the row and column index values. Then read in the file as described earlier, specifying 2 skipped lines. The TABLE array in ANSYS would look like this:

Figure 3.8 A Sample 2-D TABLE Array Dialog Box



This same example, done via command input, would look like the following:

```
*DIM,Ttx,table,4,5,,time,X-COORD
*TREAD,Ttx,t2data,txt,,2
```

Example 3: 3-D Table

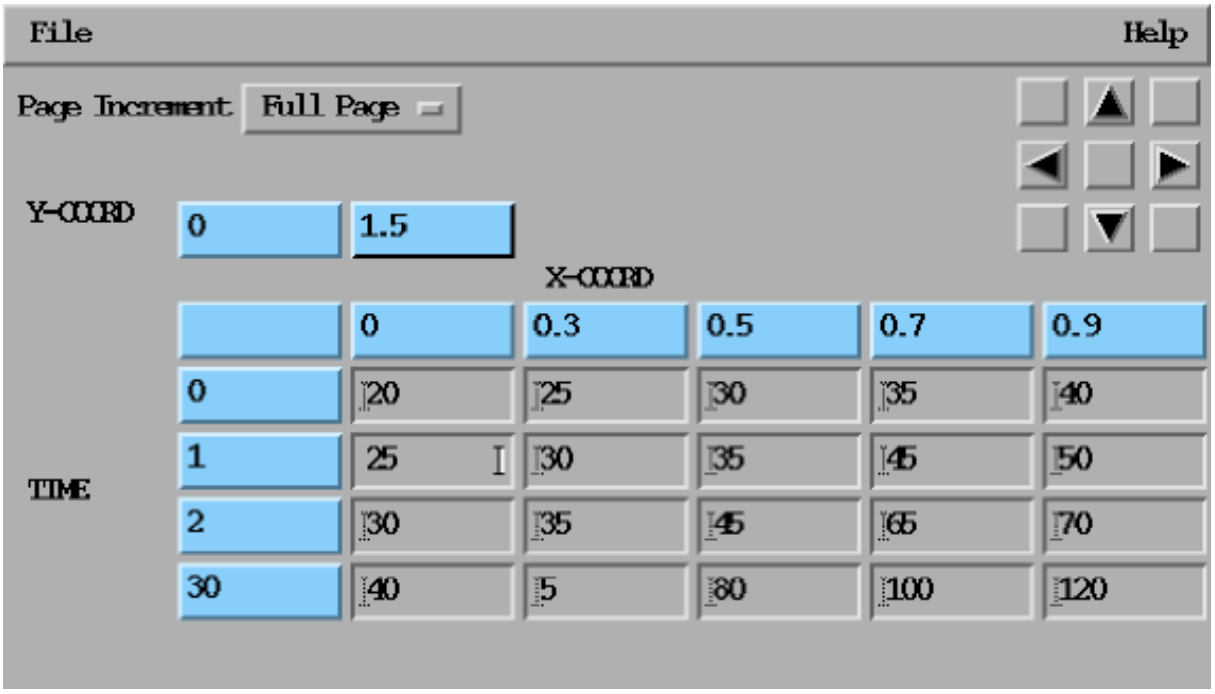
For this example, create a 3-D table named "T3data" containing temperature data as a function of time, x-coordinate, and y-coordinate and read it into a TABLE array parameter called "Ttxy." The table, in its ASCII form, would look like this:

Temp (time-X-coord) Table					
Time	X-Coordinate				
0	0	.3	.5	.7	.9
0	10	15	20	25	30
1	15	20	25	35	40
2	20	25	35	55	60
4	30	40	70	90	100
1.5	0	.3	.5	.7	.9
0	20	25	30	35	40
1	25	30	35	45	50
2	30	35	45	65	70
4	40	50	80	100	120

In the example above, the bold values (in the (0,0,Z) positions) indicate the separate planes. Each plane of data, along with the row and column index values, is repeated for the separate planes. Only the plane index value and the actual data values are different. The shaded area above shows the values that change from plane to plane.

In ANSYS, you define a TABLE parameter "Ttxy" using the ***DIM** command (**Utility Menu> Parameters> Array Parameters> Define/Edit**). In the case of a 3-D table, the table is dimensioned according to the number of rows, columns, and planes of data. The first column (TIME) is the row index values and the first row is the column index values. Specify 4 rows, 5 columns, 2 planes, row label of TIME, column label of X-COORD, and plane label of Y-COORD. Note that the data table you created has four rows and five columns of data in two planes, plus the row and column index values. Then read in the file as described earlier, specifying 2 skipped lines. The TABLE array in ANSYS would look like this for the second plane of data (Y=1.5):

Figure 3.9 A Sample 3-D TABLE Array Dialog Box



This same example, done via command input, would look like the following:

```
*DIM,Ttxy,table,4,5,2,TIME,X-COORD,Y-COORD
*TREAD,Ttxy,t3data,txt,,2
```

3.11.5.6. Interpolating Values

When accessing information from the array, ANSYS will interpolate values between those explicitly set.

As examples of how ANSYS interpolates values in TABLE arrays, consider the following:

$$A = \begin{array}{c} \\ 1.0 \\ 2.0 \\ 3.0 \end{array} \begin{array}{c} \begin{array}{c} 1.0 \end{array} \\ \begin{bmatrix} 12.0 \\ 28.0 \\ 146.4 \end{bmatrix} \end{array} \qquad PQ = \begin{array}{c} \\ 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{array} \begin{array}{c} \begin{array}{cc} 1.0 & 2.0 \end{array} \\ \begin{bmatrix} 2.8 & 4.2 \\ -9.6 & -12.3 \\ 42.0 & 9.7 \\ -4.5 & 2.0 \end{bmatrix} \end{array}$$

Given that A is a TABLE array parameter, the ANSYS program can calculate any value between A(1) and A(2), for example

- A(1.5) evaluates to 20.0 (halfway between 12.0 and 28.0)
- A(1.75) evaluates to 24.0
- A(1.9) evaluates to 26.4

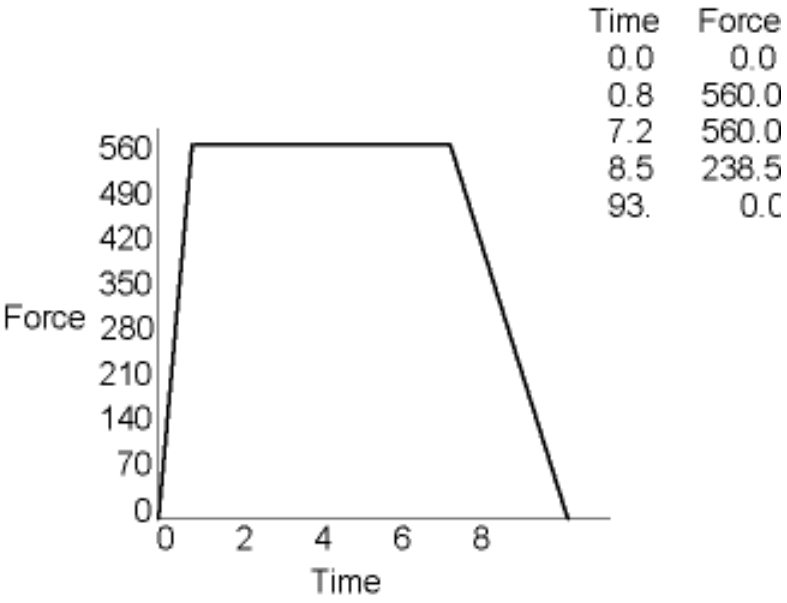
Similarly, if PQ is a TABLE array parameter

- PQ(1.5,1) evaluates to -3.4 (halfway between 2.8 and -9.6)
- PQ(1,1.5) evaluates to 3.5 (halfway between 2.8 and 4.2)

- PQ(3.5,1.3) evaluates to 14.88

This feature allows you to describe a *function*, such as y=f(x), using a TABLE array parameter. You would use the *j*=0 column for values of the independent variable x and the "regular" *j*=1 column for values of y. Consider, for example, a time-history forcing function described by five points as shown below.

Figure 3.10 Time-History Forcing Function



You can specify this function as a TABLE array parameter whose array elements are the force values, and whose row index numbers 1 through 5 are time values 0.0 through 9.3. Schematically, the parameter will then look like this:

$$\text{FORCE} = \begin{array}{c} \\ 1E-6 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 0 \\ \left[\begin{array}{c} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{array}\right. \end{array}$$

ANSYS can calculate (through linear interpolation) force values at times not specified in the FORCE parameter. For the above example, ANSYS will calculate a value of 89.4375 for FORCE(9). If a parameter location beyond the dimensions of the array is used, no extrapolation is done and the end value is used. For example, ANSYS will provide a value of 560.0 for FORCE(5,2) or 0.0 for FORCE(12)

You can see from these examples that TABLE array parameters can be very powerful tools in your analysis. Typical applications are time-history loading functions, response spectrum curves, stress-strain curves, material-versus- temperature curves, B-H curves for magnetic materials, and so forth. Be aware that TABLE array parameters require more computer time to process than the ARRAY type.

3.11.5.7. Retrieving Values into or Restoring Array Parameter Values

You can use the ***VGET** command (**Utility Menu> Parameters> Get Array Data**), which is similar to ***GET**, to retrieve ANSYS supplied values and store them in an array.

You must define a starting array location number for the array parameter the ***VGET** command creates. Looping continues over successive entity numbers for the *KLOOP* default. For example, ***VGET**,A(1),ELEM,5,CENT,X returns the centroid x-location of element 5 and stores the result in the first location of A. Retrieving continues with elements 6, 7, and so on until successive array locations are filled. In this example, if *KLOOP* is 4, then the centroid of x, y, and z are returned.

To restore array parameter values, use the ***VPUT** command (**Utility Menu> Parameters> Array Operations> Put Array Data**).

The ***VPUT** command uses the same arguments as the ***VGET** command (described above), but does the opposite of the ***VGET** operation. For a list of valid labels for ***VPUT** items, see the command's description in the *ANSYS Commands Reference*.

The ANSYS program "puts" vector items directly, without any coordinate system transformation. ***VPUT** can replace existing array items, but can't create new items. Degree of freedom results that are changed in the database are available for all subsequent operations. Other results change temporarily, and are available mainly for immediately following print and display operations.

> *Note* — Use this command with extreme caution, as it can alter entire sections of the database. The ***VPUT** command doesn't support all items on the ***VGET** item list because putting values into some locations could make the ANSYS database inconsistent.

3.11.5.8. Listing Array Parameters

As with scalar parameters, you can use the ***STATUS** command to list array parameters. The following examples illustrate the ***STATUS** command in use:

```
*STATUS
 ABBREVIATION STATUS-

  ABBREV    STRING
  SAVE_DB   SAVE
  RESUM_DB  RESUME
  QUIT      Fnc_/EXIT
  POWRGRPH  Fnc_/GRAPHICS
  ANSYSWEB  Fnc_HomePage

 PARAMETER STATUS-           (   5 PARAMETERS DEFINED)
                 (INCLUDING     2 INTERNAL PARAMETERS)

  NAME         VALUE         TYPE     DIMENSIONS
  MYCHAR      hi        CHARACTER
  MYPAR                     ARRAY       4       6       1
  MYPAR1    .987350000      SCALAR

*STATUS,XYZ(1),5,9      ! Lists rows 5 through 9 of XYZ
 PARAMETER STATUS- XYZ       (   4 PARAMETERS DEFINED)

      LOCATION          VALUE
      5     1     1   -8.98000000
      6     1     1    9.01000000
      7     1     1   -30.6000000
      8     1     1    51.0000000
      9     1     1   -51.9000000

*STATUS,FORCE(1),,,0    ! Lists parameter FORCE, includes j=0 column
```

```
PARAMETER STATUS- FORCE     (   4 PARAMETERS DEFINED)

      LOCATION          VALUE
      1     0     1   0.000000000E+00
      2     0     1   0.800000000
      3     0     1    7.20000000
      4     0     1    8.50000000
      5     0     1    9.30000000
      1     1     1   0.000000000E+00
      2     1     1    560.000000
      3     1     1    560.000000
      4     1     1    238.500000
      5     1     1   0.000000000E+00

*STATUS,T2(1,1)         ! Lists parameter T2

PARAMETER STATUS- T2        (   4 PARAMETERS DEFINED)

      LOCATION          VALUE
      1     1     1   0.600000000
      2     1     1    2.00000000
      3     1     1   -1.80000000
      4     1     1    4.00000000
      1     2     1    7.00000000
      2     2     1    5.00000000
      3     2     1    9.10000000
      4     2     1    62.5000000
      1     3     1   2.000000000E-04
      2     3     1   -3.50000000
      3     3     1    22.0000000
      4     3     1   1.000000000E-02

*STATUS,RESULT(1)!Lists parameter RESULT


PARAMETER STATUS- RESULT     ( 4 PARAMETERS DEFINED)

LOCATION     VALUE
1     1     1     SX(CHAR)
2     1     1     SY(CHAR)
3     1     1     SZ(CHAR)
```

3.11.6. Writing Data Files

You can write formatted data files (tabular formatting) from data held in arrays through the ***VWRITE** command. The command takes up to 10 array vectors as arguments and writes the data contained in those vectors to the currently open file (***CFOPEN** command). The format for each vector is specified with FORTRAN 77 data descriptors on the line following the ***VWRITE** command (therefore you can't issue the ***VWRITE** command from the ANSYS input window.)

An array vector, specified with a starting element location (such as MYARRAY(1,2,1)). You can also use an expression, which is evaluated as a constant value for that field in each row of the data file. The keyword SEQU evaluates to a sequential column of integers, starting from one.

The format of each row in the data file is determined by the data descriptor line. You must include one descriptor for each argument to the command. Do not include the word FORMAT in the descriptor line. You can use any real format or character format descriptor; however, you may not use either integer or list directed descriptors.

3.11.6.1. Format Data Descriptors

If you aren't familiar with FORTRAN data descriptors, this section will get you started with formatting your data file. For more information, consult the documentation for the FORTRAN 77 compiler for your particular platform.

You must provide a data descriptor for each data item you specify as an argument to the ***VWRITE** command. In general, you can use the F descriptor (floating point) for any numeric values. The F descriptor takes the syntax

F*w.d*

where

w
Is the width of the data field in characters.

d
Is the number of digits to the right of the decimal point.

Thus, for a field that is 10 characters wide and has eight characters after the decimal point, you would use the following data descriptor:

```
F10.8
```

For character fields, you can use the A descriptor. The A descriptor has the syntax

A*w*

where

w
Is the width of the data field in characters.

Thus, for a character field that is eight characters wide, the descriptor is

```
A8
```

The following examples illustrate the ***VWRITE** command and data descriptors in use.

Given that the MYDATA array has been dimensioned and filled with the following values:

$$\text{MYDATA} = \begin{bmatrix} 2.15215183 & 3.89075020 & 5.28636971 & 7.15706483 & 13.7859423 & 87.4970443 \\ 2.30485343 & 4.44486730 & 5.40919563 & 7.68192625 & 15.5483820 & 86.5677915 \\ 2.01051819 & 3.39152436 & 5.93663807 & 7.38584253 & 18.4635868 & 45.7263566 \\ 2.36833012 & 3.32711472 & 5.63220341 & 7.22482004 & 18.7977889 & 39.7902425 \\ 2.84819512 & 4.76350638 & 5.97802354 & 7.29258882 & 14.8096356 & 62.0843906 \\ 2.22795343 & 3.48214546 & 5.54685145 & 7.90325139 & 14.0708891 & 37.6009897 \end{bmatrix}$$

The following short macro first defines the scalar parameter X as having a value of 25 and then opens the file **vector** (***CFOPEN** command). The ***VWRITE** command then defines the data to be written to the file. In this case, the first vector written uses the SEQU keyword to provide row numbers. Note that in some cases that constants, scalar parameters, and operations that include array element values are written to the file. Note the data file contents for these items.

```
x=25
*cfopen,vector
*vwrite,SEQU,mydata(1,1,1),mydata(1,2,1),mydata(1,3,1),10.2,x,mydata(1,1,1)+3
(F3.0,'  ',F8.4,'  ',F8.1,'  'F8.6,'  ',F4.1,'  'F4.0,'  'F8.1)
*cfclos
```

The macro creates the following data file:

```
  1.    2.1522       3.9  5.286370  10.2   25.       5.2
 2.    2.3049       4.0  5.409196  10.2   25.       5.2
 3.    2.0105       3.4  5.936638  10.2   25.       5.2
 4.    2.3683       3.3  5.632203  10.2   25.       5.2
```

```
 5.    2.8491       4.8  5.978024  10.2   25.       5.2
 6.    2.2280       3.5  5.546851  10.2   25.       5.2
```

The second example uses the following previously dimensioned and filled array:

$$\text{MYDATA} = \begin{bmatrix} 10 & 50 \\ 20 & 70 \\ 30 & 80 \end{bmatrix}$$

Note the use of descriptors in the following example ***VWRITE** command:

```
*vwrite,SEQU,mydata(1,1),mydata(1,2),(mydata1(1,1)+mydata1(1,2))
(' Row',F3.0,' contains ',2F7.3,'. Is their sum ',F7.3,' ?')
```

The resulting data file is

```
Row 1. contains  10.000 50.000.    Is their sum  60.000 ?
Row 2. contains  20.000 60.000.    Is their sum  60.000 ?
Row 3. contains  30.000 70.000.    Is their sum  60.000 ?
```

3.11.7. Operations Among Array Parameters

Just as parametric expressions and functions allow operations among scalar parameters, a series of commands is available to perform operations among array parameters. There are classes of operations: operations on columns (vectors), known as *vector operations* and operations on entire matrices (arrays), known as *matrix operations*. All operations are affected by a set of specification commands, which are discussed in Section 3.11.7.3: Specification Commands for Vector and Matrix Operations.

3.11.7.1. Vector Operations

Vector operations are simply a set of operations - addition, subtraction, sine, cosine, dot product, cross product, etc. - repeated over a sequence of array elements. Do-loops (discussed in Section 4.5.5: Looping: Do-Loops) can be employed for this purpose, but a more convenient and much faster way is to use the vector operation commands - ***VOPER**, ***VFUN**, ***VSCFUN**, ***VITRP**, ***VFILL**, ***VREAD**, and ***VGET**. Of these listed vector operation commands, only ***VREAD** and ***VWRITE** are valid for character array parameters. Other vector operation commands apply only to array parameters dimensioned (***DIM**) as ARRAY type or TABLE type.

The ***VFILL**, ***VREAD**, ***VGET**, ***VWRITE**, and ***DIM** commands were introduced earlier in this chapter. Other commands that are discussed in this section include

***VOPER** or **Utility Menu> Parameters> Array Operations> Vector Operations**
Performs an operation on two input array vectors and produces a single output array vector.

***VFUN** or **Utility Menu> Parameters> Array Operations> Vector Functions**
Performs a function on a single input array vector and produces a single output array vector.

***VSCFUN** or **Utility Menu> Parameters> Array Operations> Vector-Scalar Func**
Determines the properties of a single input array vector and places the result in a specified scalar parameter.

***VITRP** or **Utility Menu> Parameters> Array Operations> VectorInterpolate**
Forms an array parameter (type ARRAY) by interpolating an array parameter (type TABLE) at specified table index locations.

The examples below illustrate the use of some of these commands. Refer to the *ANSYS Commands Reference* for syntactical information about these commands. For all of the following examples, the array parameters (of type ARRAY) X, Y, and THETA have been dimensioned and defined.

$$X = \begin{bmatrix} -2 & 6 & 8 & 0 \\ 1 & 0 & 2 & 12 \\ 4 & -3 & -1 & 7 \\ -8 & 1 & 10 & -5 \end{bmatrix} \qquad Y = \begin{bmatrix} 3 & 2 & 5 & -6 \\ -5 & -7 & 1 & 0 \\ 8 & 0 & 0 & 11 \\ 1 & 4 & 9 & 16 \end{bmatrix}$$

$$THETA = \begin{bmatrix} 0 \\ 15 \\ 30 \\ 45 \\ 60 \\ 75 \\ 90 \end{bmatrix}$$

In the following example, the result array is first dimensioned (Z1). The ***VOPER** command then adds column 2 of X to column 1 of Y, both starting at row 1, and then places the result into Z1. Notice that the starting location (the row and column index numbers) must be specified for all array parameters. The operation then progresses sequentially down the specified vector.

```
*DIM,Z1,ARRAY,4
*VOPER,Z1(1),X(1,2),ADD,Y(1,1)
```

$$Z1 = \begin{bmatrix} 9 \\ -5 \\ 5 \\ 2 \end{bmatrix}$$

In the following example, again the result array (Z2) is dimensioned first. The ***VOPER** command then multiplies the first column of X (starting at row 2) with the fourth column of Y (starting at row 1) and writes the results to Z2 (starting at row 1).

```
*DIM,Z2,ARRAY,3
*VOPER,Z2(1),X(2,1),MULT,Y(1,4)
```

$$Z2 = \begin{bmatrix} -6 \\ 0 \\ -88 \end{bmatrix}$$

In this example, again the results array (Z4) is dimensioned first. The ***VOPER** command then performs the cross product of four pairs of vectors, one pair for each row of X and Y. The *i*, *j*, and *k* components of these vectors are columns 1, 2, and 3 respectively of X and columns 2, 3, and 4 of Y. The results are written to Z4, whose *i*, *j*, and *k* components are vectors 1, 2, and 3 respectively.

```
*DIM,Z4,ARRAY,4,3
*VOPER,Z4(1,1),X(1,1),CROSS,Y(1,2)
```

$$Z4 = \begin{bmatrix} -76 & 4 & -22 \\ -2 & -14 & 1 \\ -33 & -44 & 0 \\ -74 & 168 & -76 \end{bmatrix}$$

In the following example, the results array (A3) is dimensioned first. The ***VFUN** command then raises each element in vector 2 of X to the power of 2 and writes the results to A3.

```
*DIM,A3,ARRAY,4
*VFUN,A3(1),PWR,X(1,2),2
```

$$A3 = \begin{bmatrix} 36 \\ 0 \\ 9 \\ 1 \end{bmatrix}$$

In this example, the results array (A4) is dimensioned. The two ***VFUN** commands then calculate the cosine and sine of array elements in THETA and place the results in the first and second columns, respectively, of A4. Notice that A4 now represents a circular arc spanning 90°, described by seven points (whose x, y, and z global Cartesian coordinates are the three vectors). The arc has a radius of 1.0 and lies parallel to the x-y plane at z = 2.0.

```
*DIM,A4,ARRAY,7,3
*AFUN,DEG
*VFUN,A4(1,1),COS,THETA(1)
*VFUN,A4(1,2),SIN,THETA(1)
A4(1,3)=2,2,2,2,2,2,2
```

$$A4 = \begin{bmatrix} 1.0 & 0.0 & 2.0 \\ 0.966 & 0.259 & 2.0 \\ 0.866 & 0.5 & 2.0 \\ 0.707 & 0.707 & 2.0 \\ 0.5 & 0.866 & 2.0 \\ 0.259 & 0.966 & 2.0 \\ 0.0 & 1.0 & 2.0 \end{bmatrix}$$

In this example, the results array (A5) is first dimensioned. Then, the ***VFUN** command calculates the tangent vector at each point on the curve represented by A4, normalizes it to 1.0, and places the results in A5.

```
*DIM,A5,ARRAY,7,3
*VFUN,A5(1,1),TANG,A4(1,1)
```

$$A5 = \begin{bmatrix} -0.131 & 0.991 & 0 \\ -0.259 & 0.965 & 0 \\ -0.5 & 0.866 & 0 \\ -0.707 & 0.707 & 0 \\ -0.866 & 0.5 & 0 \\ -0.966 & 0.259 & 0 \\ -0.991 & 0.131 & 0 \end{bmatrix}$$

Two additional ***VOPER** operations, gather (GATH) and scatter (SCAT), are used to copy values from one vector to another based on numbers contained in a "position" vector. The following example demonstrates the gather operation. Note that, as always, the results array must be dimensioned first. In the example, the gather operation copies the value of B1 to B3 (using the index positions specified in B2). Note that the last element in B3 is 0 as this is its initialized value.

```
*DIM,B1,,4
*DIM,B2,,3
*DIM,B3,,4
B1(1)=10,20,30,40
B2(1)=2,4,1
*VOPER,B3(1),B1(1),GATH,B2(1)
```

$$B3 = \begin{bmatrix} 20 \\ 40 \\ 10 \\ 0 \end{bmatrix}$$

3.11.7.2. Matrix Operations

Matrix operations are mathematical operations between numerical array parameter matrices, such as matrix multiplication, calculating the transpose, and solving simultaneous equations.

Commands discussed in this section include

***MOPER** or **Utility Menu> Parameters> Array Operations> Matrix Operations**
Performs matrix operations on two input array parameter matrices and produces one output array parameter matrix. Matrix operations include:

- Matrix multiplication
- Solution of simultaneous equations
- Sorting (in ascending order) on a specified vector in a matrix
- Covariance between two vectors
- Correlation between two vectors

***MFUN** or **Utility Menu> Parameters> Array Operations> Matrix Functions**
Copies or transposes an array parameter matrix (accepts one input matrix and produces one output matrix).

***MFOURI** or **Utility Menu> Parameters> Array Operations> Matrix Fourier**
Calculates the coefficients for or evaluates a Fourier series.

The examples below illustrate the use of some of these commands. Refer to the *ANSYS Commands Reference* for syntactical information about these commands.

This example shows the sorting capabilities of the ***MOPER** command. For this example, assume that the array (SORTDATA) has been dimensioned and its element values have been defined as follows:

$$\text{SORTDATA} = \begin{bmatrix} 3 & 10 & 11 \\ 5 & -4 & 12 \\ 8 & -9 & 13 \\ 2 & 7 & 14 \\ 6 & 1 & 15 \end{bmatrix}$$

First, the OLDORDER array is dimensioned. The ***MOPER** command will place the original order of the rows into OLDORDER. The ***MOPER** command then sorts the rows in SORTDATA so that the 1,1 vector is now in ascending order.

```
*dim,oldorder,,5
*moper,oldorder(1),sortdata(1,1),sort,sortdata(1,1)
```

The following array values result from the ***MOPER** command:

$$\text{SORTDATA} = \begin{bmatrix} 2 & 7 & 14 \\ 3 & 10 & 11 \\ 5 & -4 & 12 \\ 6 & 1 & 15 \\ 8 & -9 & 13 \end{bmatrix} \qquad \text{OLDORDER} = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 3 \end{bmatrix}$$

To put the SORTDATA array back into its original order, you could then issue the following command:

```
*moper,oldorder(1),sortdata(1,1),sort,oldorder(1,1)
```

In the following example, the ***MOPER** command solves a set of simultaneous equations. The following two arrays have been dimensioned and their values assigned:

$$A = \begin{bmatrix} 2 & 4 & 3 & 2 \\ 3 & 6 & 5 & 2 \\ 2 & 5 & 2 & -3 \\ 4 & 5 & 14 & 14 \end{bmatrix} \qquad B = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 11 \end{bmatrix}$$

The ***MOPER** command can solve a set of simultaneous equations for a square matrix. The equations take the form

$a_{n1}X_1 + a_{n2}X_2 + , \ldots , + a_{nn}X_n = b_n$

In the case of the above arrays, the ***MOPER** command will solve the following set of simultaneous equations:

$2X_1 + 4X_2 + 3X_3 + 2X_4 = 2$

$3X_1 + 6X_2 + 5X_3 + 2X_4 = 2$

$2X_1 + 5X_2 + 2X_3 - 3X_4 = 3$

$4X_1 + 5X_2 + 14X_3 + 14X_4 = 11$

To solve the equations, first the results array (C) is dimensioned. Then the ***MOPER** command solves the equations, using A as the matrix of *a* coefficients and B as a vector of *b* values.

```
*DIM,C,,4
*MOPER,C(1),A(1,1),SOLV,B(1)
```

The C array now contains the following solutions.

$$C = \begin{bmatrix} -66 \\ 26 \\ 6 \\ 4 \end{bmatrix}$$

The following example shows the ***MFUN** command used to transpose data in an array. For this example, assume that the array (DATA) was dimensioned and filled with the following values:

$$DATA = \begin{bmatrix} 34 & 25 \\ 22 & 68 \\ -7 & 12 \end{bmatrix}$$

As always, the results array (DATATRAN) is dimensioned first, then the ***MFUN** command transposes the values and writes them to DATATRAN.

```
*DIM,DATATRAN,,2,3
*MFUN,DATATRAN(1,1),TRAN,DATA(1,1)
```

The following shows the results in the DATATRAN array:

$$DATATRAN = \begin{bmatrix} 34 & 22 & -7 \\ 25 & 68 & 12 \end{bmatrix}$$

3.11.7.3. Specification Commands for Vector and Matrix Operations

All the vector and matrix operation commands are affected by the setting of the following specification commands: ***VCUM**, ***VABS**, ***VFACT**, ***VLEN**, ***VCOL**, and***VMASK**. (Of all specification commands, only ***VLEN** and ***VMASK**, in conjunction with ***VREAD** or ***VWRITE**, are valid for character array parameters.) You can check the status of these commands with the ***VSTAT** command. Most of these commands (and their corresponding GUI paths) were introduced earlier in this chapter. The others are explained in the following.

With the exception of the ***VSTAT** command, which you cannot access directly in the GUI, all of the specification commands described below are available via menu path **Utility Menu> Parameters> Array Operations> Operation Settings**.

Important: All specification commands are reset to their default settings after each vector or matrix operation.

The following lists the available array specification commands:

***VCUM**
Specifies whether results will be cumulative or noncumulative (overwriting previous results). *ParR*, the result of a vector operation, is either added to an existing parameter of the same name or overwritten. The default is noncumulative results, that is, *ParR* overwrites an existing parameter of the same name.

***VABS**
Applies an absolute value to any or all of the parameters involved in a vector operation. The default is to use the real (algebraic) value.

***VFACT**
Applies a scale factor to any or all of the parameters involved in a vector operation. The default scale factor is 1.0 (full value).

***VCOL**
Specifies the number of columns in matrix operations. The default is to fill all locations of the result array from the specified starting location.

***VSTAT**
Lists the current specifications for the array parameters.

***VLEN** or **Utility Menu> Parameters> Array Operations> Operation Settings**
Specifies the number of rows to be used in array parameter operations.

***VMASK** or **Utility Menu> Parameters> Array Operations> Operation Settings**
Specifies an array parameter as a masking vector.

The following table lists the various specification commands and the vector and matrix array commands that they affect.

	***VABS**	***VFACT**	***VCUM**	***VCOL**	***VLENNROW,NINC**		***VMASK**
***MFOURI**	No	No	No	No	No	No	No
***MFUN**	Yes	Yes	Yes	No	Yes	No	Yes
***MOPER**	Yes	Yes	Yes	No	Yes	No	Yes
***VFILL**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VFUN**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VGET**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VITRP**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VOPER**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
VPLOT	No	No	N/A	N/A	Yes	Yes	Yes
***VPUT**	Yes	Yes	No	N/A	Yes	Yes	Yes
***VREAD**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VSCFUN**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VWRITE**	No	No	N/A	N/A	Yes	Yes	Yes

The examples below illustrate the use of some of the specification commands. Refer to the *ANSYS Commands Reference* for syntactical information about these commands.

In the following, the results array (CMPR) is dimensioned. The two ***VFUN** commands, in conjunction with the preceding ***VMASK** and ***VLEN** commands, then compress selected data and write them to specified locations in CMPR. The complement to the COMP operation is the EXPA operation on the ***VFUN** command.

```
*DIM,CMPR,ARRAY,4,4
*VLEN,4,2! Do next *V---- operation on four rows,
! skipping every second row
```

```
*VFUN,CMPR(1,2),COMP,Y(1,1)
*VMASK,X(1,3)!Use column 3 of X as a mask for next *V----
! operation
*VFUN,CMPR(1,3),COMP,Y(1,2)
```

$$CMPR = \begin{bmatrix} 0 & 3 & 2 & 0 \\ 0 & 8 & -7 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

This example uses the ***VFACT** command to round the values in an array vector to the number of decimal places specified by the NUMDP scalar parameter (set to 2 in the example). The NUMDATA array has been dimensioned and filled with the following values:

$$NUMDATA = \begin{bmatrix} 2.526 \\ 2.524 \\ -6.526 \\ -6.524 \end{bmatrix}$$

```
numdp=2
*vfact,10**numdp
*vfun,numdata(1),copy,numdata(1)
*vfun,numdata(1),nint,numdata(1)
*vfact,10**(-numdp)
*vfun,numdata(1),copy,numdata(1)
```

or, you can use a slightly shorter version

```
numdp=2
*vfact,10**numdp
*vfun,numdata(1),copy,numdata(1)
*vfact,10**(-numdp)
*vfun,numdata(1),nint,numdata(1)
```

The resultant NUMDATA array is then:

$$NUMDATA = \begin{bmatrix} 2.53 \\ 2.52 \\ -6.53 \\ -6.52 \end{bmatrix}$$

This example uses the ***VLEN** and ***VMASK** commands to find the set of prime numbers less than 100. An array, MASKVECT, is created using 1.0 to indicate that the row value is a prime number and 0.0 to indicate that the value isn't prime. The algorithm used to create the mask vector is to initialize all rows whose value is greater than 1 to 1.0 and then loop through the range of possible factors, eliminating all multiples of the factor. The ***VLEN** command sets the row increment for performing operations to FACTOR. When the ***VFILL** command is processed, the row number is incremented by this value. Because the starting row is FACTOR x 2, the rows are processed by each loop in the following manner: FACTOR x 2, FACTOR x 3, FACTOR x 4, etc.

```
*dim,maskvect,,100
*vfill,maskvect(2),ramp,1
*do,factor,2,10,1
*vlen,,factor
*vfill,maskvect(factor*2),ramp,0
```

```
*enddo
*vmask,maskvect(1)
*dim,numbers,,100
*vfill,numbers(1),ramp,1,1
*status,numbers(1),1,10
```

The resultant output from the ***STATUS** command, showing the first 10 elements in NUMBERS is:

```
PARAMETER STATUS- NUMBERS   (   5 PARAMETERS DEFINED)
                  (INCLUDING     2 INTERNAL PARAMETERS)

      LOCATION          VALUE
      1     1     1   0.000000000E+00
      2     1     1   2.00000000
      3     1     1   3.00000000
      4     1     1   0.000000000E+00
      5     1     1   5.00000000
      6     1     1   0.000000000E+00
      7     1     1   7.00000000
      8     1     1   0.000000000E+00
      9     1     1   0.000000000E+00
     10     1     1   0.000000000E+00
```

3.11.8. Plotting Array Parameter Vectors

You can graphically display array vector values using the ***VPLOT** command.

The following demonstrates some of the capabilities of the ***VPLOT** command. For this example, two TABLE arrays (TABLEVAL and TABLE) and one numeric array have been dimensioned and filled with the following values:

$$\text{TABLEVAL} = \begin{array}{c} \\ 4 \\ 7 \\ 15 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 3 \quad 9 \end{array} \\ \begin{bmatrix} 6 & 12 \\ 8 & 6 \\ 10 & 3 \end{bmatrix} \end{array} \qquad \text{TABLE2} = \begin{array}{c} \\ 19 \\ 88 \\ 99 \end{array} \begin{array}{c} \begin{array}{c} 0 \quad 40 \end{array} \\ \begin{bmatrix} 70 \\ 80 \\ 95 \end{bmatrix} \end{array}$$

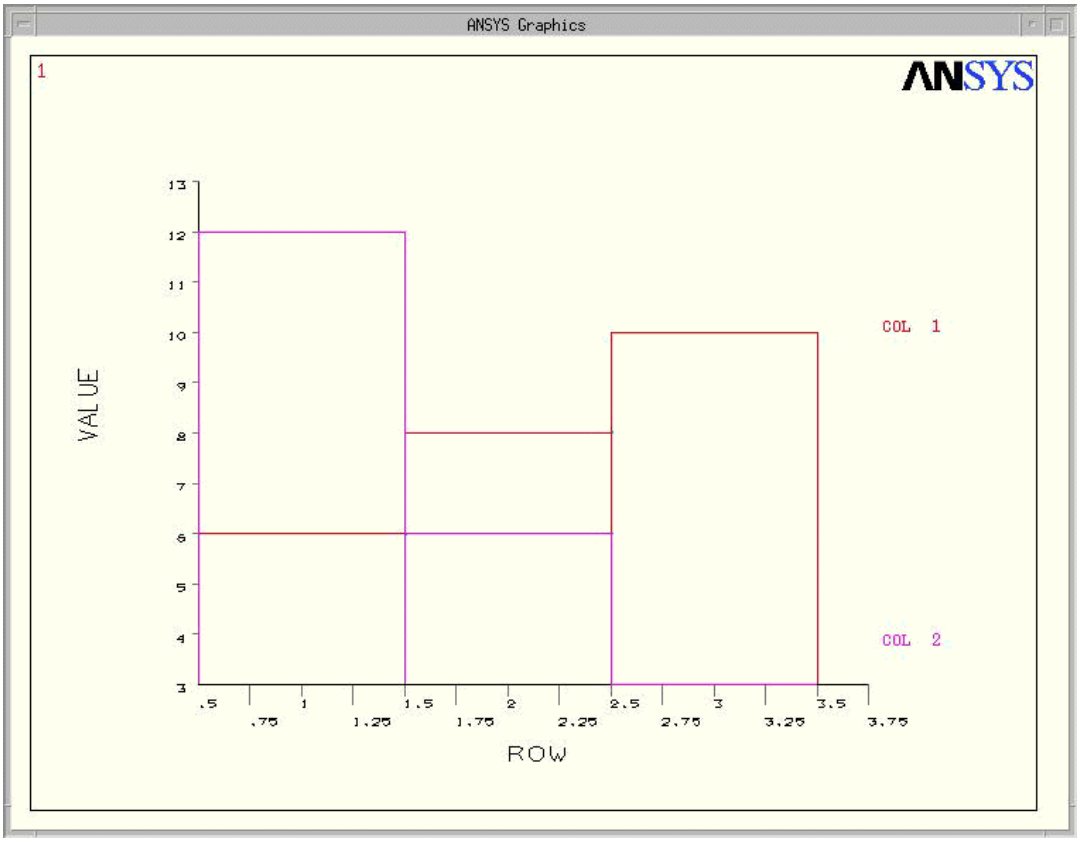
$$\text{ARRAYVAL} = \begin{bmatrix} 6 & 12 \\ 8 & 6 \\ 10 & 3 \end{bmatrix}$$

The following are example ***VPLOT** commands and their resulting plots. Note that since ARRAY data is unordered it is plotted as a histogram; TABLE data is ordered and is therefore plotted as a curve.

The plot (below) resulted from the following command.

```
*vplot,,arrayval(1,1),2
```

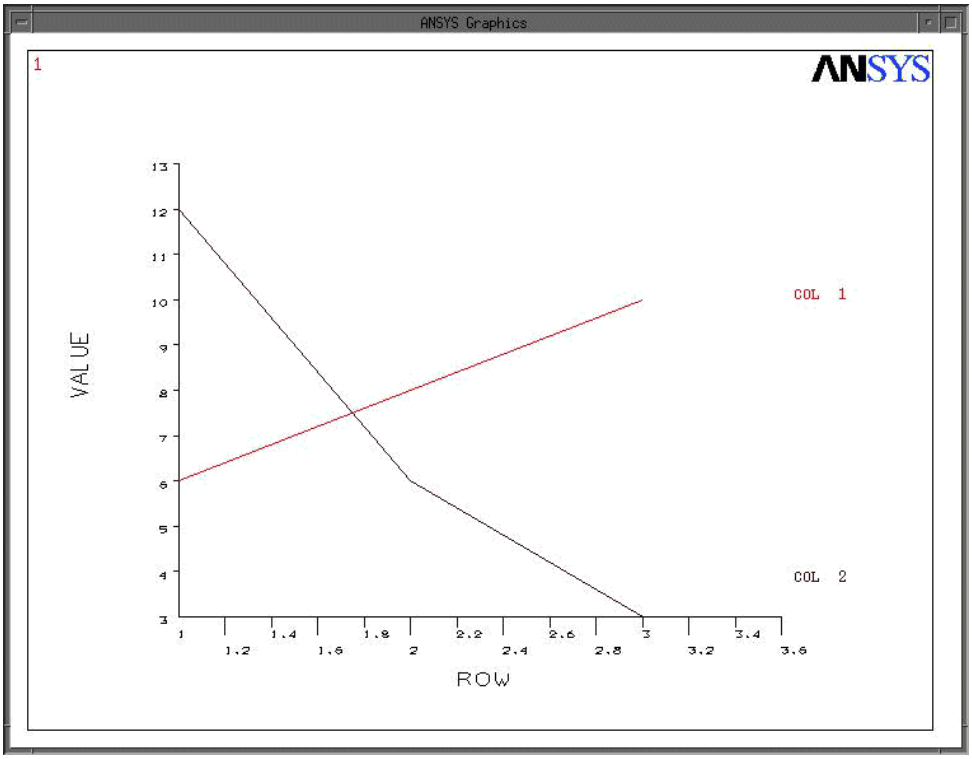
Figure 3.11 Sample Plot



The plot (below) resulted from the following command.

```
*vplot,,tableval(1,1),2
```

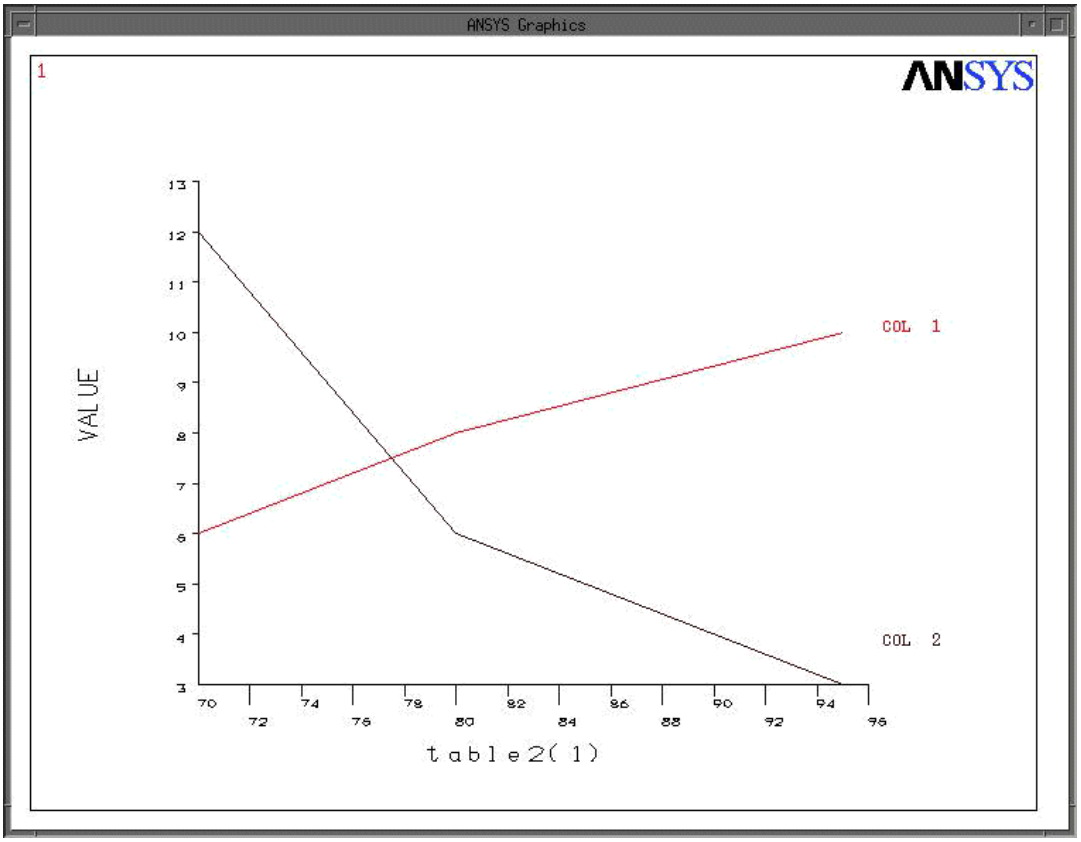
Figure 3.12 Sample Plot



The plot (below) resulted from the following command.

```
*vplot,table2(1),tableval(1,1),2
```

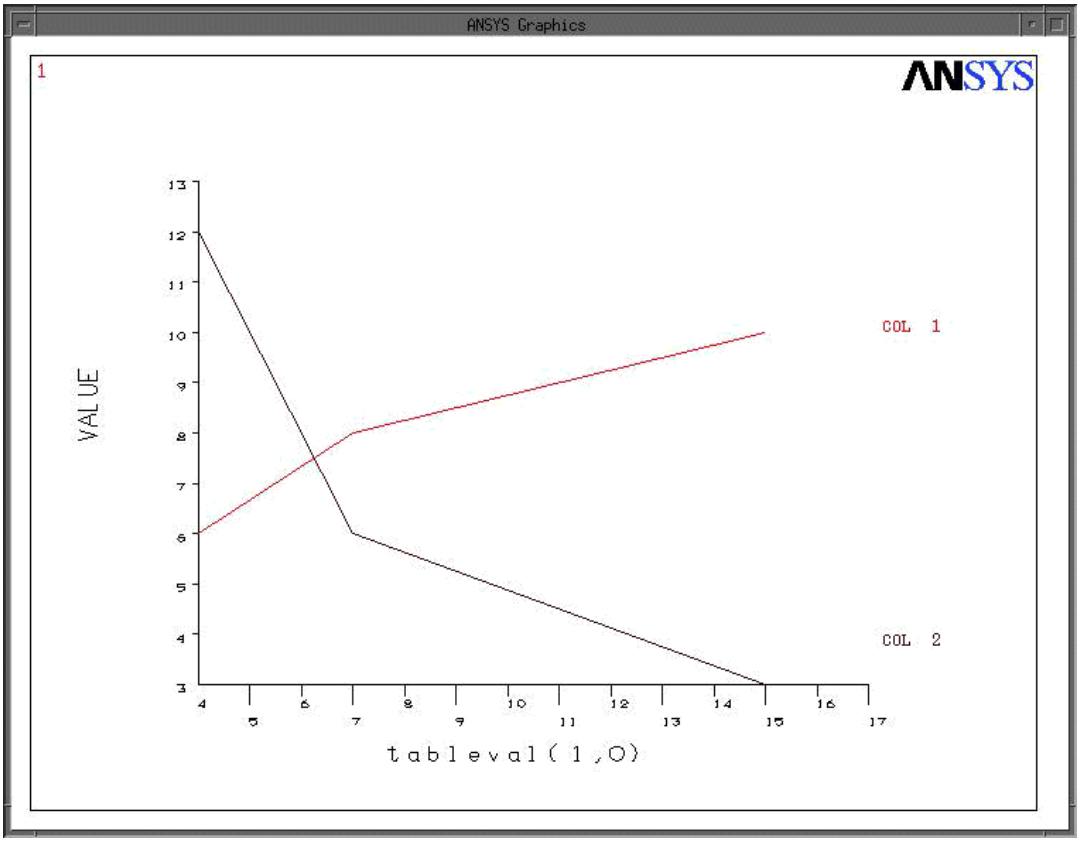
Figure 3.13 Sample Plot



The plot (below) resulted from the following command.

```
*vplot,tableval(1,0),tableval(1,1),2
```

Figure 3.14 Sample Plot



3.11.9. Modifying Curve Labels

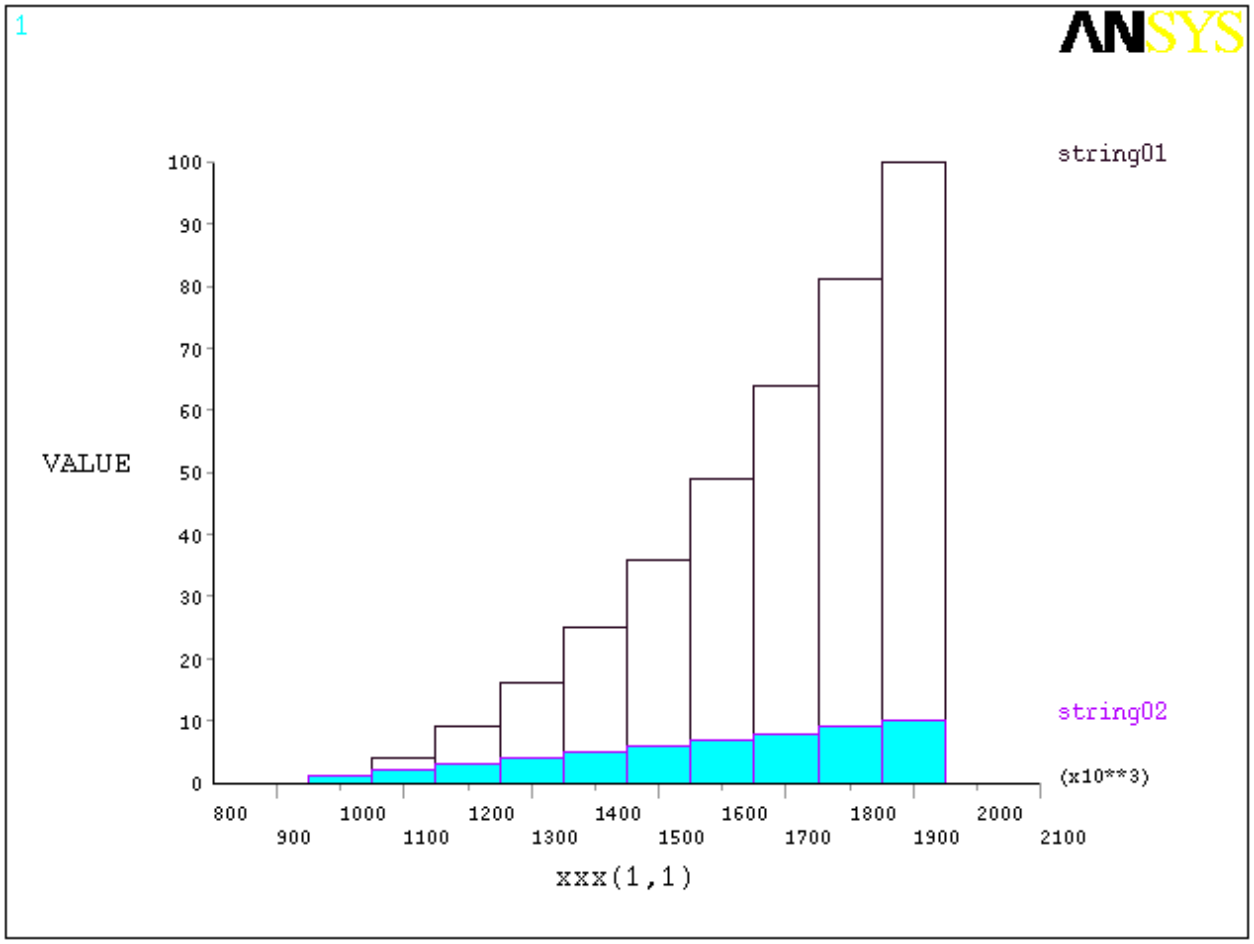
When you use ***VPLOT** to create your curves, default labels are assigned. Normally, the label for curve 1 is "COL 1", the label for curve 2 is "COL 2" and so on; the column number is the field containing the dependent variables for that particular curve. You can use the **/GCOLUMN** command to apply your own labels to the curves (any string of up to eight characters).

The example below uses the **/GCOLUMN** command at the beginning of the program input to apply the labels "string01" and "string02" to the array curve.

```
/gcol,1,string01
/gcol,2,string02

*dim,xxx,array,10
*dim,yyy,array,10,2

xxx( 1,1) =1e6
xxx( 2,1) = 1e6 + 1e5
xxx( 3,1) = 1e6 + 2e5
xxx( 4,1) = 1e6 + 3e5
xxx( 5,1) = 1e6 + 4e5
xxx( 6,1) = 1e6 + 5e5
xxx( 7,1) = 1e6 + 6e5
xxx( 8,1) = 1e6 + 7e5
xxx( 9,1) = 1e6 + 8e5
xxx(10,1) = 1e6 + 9e5

yyy( 1,1) = 1
yyy( 2,1) = 4
yyy( 3,1) = 9
yyy( 4,1) = 16
yyy( 5,1) = 25
```

```
yyy( 6,1) = 36
yyy( 7,1) = 49
yyy( 8,1) = 64
yyy( 9,1) = 81
yyy(10,1) = 100

yyy( 1,2) = 1
yyy( 2,2) = 2
yyy( 3,2) = 3
yyy( 4,2) = 4
yyy( 5,2) = 5
yyy( 6,2) = 6
yyy( 7,2) = 7
yyy( 8,2) = 8
yyy( 9,2) = 9
yyy(10,2) = 10

*vplo,xxx(1,1), yyy(1,1) ,2
```

Figure 3.15 Sample Plot With User-specified Labels



The labels can be returned to the default value (COL 1 and COL 2) by issuing the **/GCOLUMN** command with no string specified.

```
/gcol,1
/gcol,2
```

Chapter 4: APDL as a Macro Language

4.1. What is an APDL Macro?

You can record a frequently used sequence of ANSYS commands in a macro file (these are sometimes called command files). Creating a macro enables you to, in effect, create your own custom ANSYS command. For example, calculating power loss due to eddy currents in a magnetic analysis would require a series of ANSYS commands in the postprocessor. By recording this set of commands in a macro, you have a new, single command that executes all of the commands required for that calculation. In addition to executing a series of ANSYS commands, a macro can call GUI functions or pass values into arguments.

You can also nest macros. That is, one macro can call a second macro, the second macro can call a third macro, and so on. You can use up to 20 nesting levels, including any file switches caused by the ANSYS **/INPUT** command. After each nested macro executes, the ANSYS program returns control to the previous macro level.

The following is a very simple example macro file. In this example, the macro creates a block with dimensions 4, 3, and, 2 and a sphere with a radius of 1. It then subtracts the sphere from one corner of the block.

```
/prep7
/view,,-1,-2,-3
block,,4,,3,,2
sphere,1
vsbv,1,2
finish
```

If this macro were called **mymacro.mac**, you could execute this sequence of commands with the following single ANSYS command

```
*use,mymacro
```

or (because the extension is **.mac**)

```
mymacro
```

Although this is not a realistic macro, it does illustrate the principle.

This chapter provides information on the various ways you can create, store, and execute macros. It also discusses the basic information you need to use APDL as a scripting language in creating macros.

4.2. Creating a Macro

You can create macros either within ANSYS itself or using your text editor of choice (such as emacs, vi, or wordpad). If your macro is fairly simple and short, creating it in ANSYS can be very convenient. If you are creating a longer, more complex macro or editing an existing macro then you will need a text editor. Also, using a text editor allows you to use a similar macro or ANSYS log file as the source for your macro.

For any long, complex macro you should always consider either using a similar macro as a starting point or running the task interactively in ANSYS and using the resulting log file as the basis of your macro. Either method can greatly reduce the time and effort required to create a suitable macro.

4.2.1. Macro File Naming Conventions

Macros are a sequence of ANSYS commands stored in a file. Macros should not have the same name as an existing ANSYS command, or start with the first four characters of an ANSYS command, because ANSYS will execute the internal command instead of the macro. The following naming restrictions apply to macro files:

- The file name cannot exceed 32 characters.
- The file name cannot begin with a numeral.
- The file extension cannot contain more than eight characters (if you are executing the macro as if it were an ANSYS command it should have the extension **.mac**.)
- The file name or extension cannot contain spaces.
- The file name or extension cannot contain any characters prohibited by your file system and for portability should not contain any characters prohibited by either UNIX or Windows file systems.

To ensure that you are not using the name of an ANSYS command, before creating a macro try running the file name that you wish to use as an ANSYS command. If ANSYS returns the message shown below, you will know that the command is not used in the current processor. You should check the macro file name in each processor in which you plan to use the macro. (You could also check if the macro file name matches any command listed in the online documentation; however, this method cannot locate the names of undocumented commands.)

Figure 4.1 ANSYS Message Box for Unknown Command

Using the **.mac** extension allows ANSYS to execute the macro as it would any internal command. You should avoid using the extension **.MAC** because it is used for ANSYS internal macros.

4.2.2. Macro Search Path

By default, ANSYS searches for a user macro file (**.mac** extension) in the following locations:

1. The **/ansys_inc/v81/ansys/apdl** directory.
2. The directory (or directories) designated by the **ANSYS_MACROLIB** environment variable (if defined) or the login (home) directory. This environment variable is documented in The ANSYS Environment chapter of the *ANSYS Operations Guide*.
3. The directory designated by the **$HOME** environment variable.
4. The working directory.

You can place macros for your personal use in your home directory. Macros that should be available across your site should be placed in the **/ansys_inc/v81/ansys/apdl** directory or some commonly accessible directory that everyone can reference through the **ANSYS_MACROLIB** environment variable.

For Windows users: The "current directory" is the default directory (usually a network resource) set by administrators and you should ask your network administrator for its location. You can use environment variables to create a local "home directory." The local home directory is checked after the default directory designated in your domain profile.

4.2.3. Creating a Macro Within ANSYS

You can create a macro by four methods from within ANSYS:

- Issue the ***CREATE** command in the input window. Parameter values are not resolved and parameter names are written to the file.
- Use the ***CFOPEN**, ***CFWRITE**, and ***CFCLOS** commands. Parameter names are resolved to their current values and those values are written to the macro file.
- Issue the **/TEE** command in the input window. This command writes a list of commands to a file at the same time that the commands are being executed. As the commands are executed in the current ANSYS session, parameter names are resolved to their current values. However, in the file that is created, parameter values are not resolved and parameter names are written instead.
- Choose the **Utility Menu> Macro> Create Macro** menu item. This method opens a dialog box that can be used as a simple, multiline editor for creating macros. Parameter values are not resolved and parameter names are written to the file.

The following sections detail each of these methods.

4.2.3.1. Using *CREATE

Issuing ***CREATE** redirects ANSYS commands entered in the command input window to the file designated by the command. All commands are redirected until you issue the ***END** command. If an existing file has the same name as the macro file name you specify, the ANSYS program overwrites the existing file.

For example, suppose that you want to create a macro called **matprop.mac**, which automatically defines a set of material properties. The set of commands entered into the input window for this macro might look like this:

```
*CREATE,matprop,mac,macros
MP,EX,1,2.07E11
MP,NUXY,1,.27
MP,DENS,1,7835
MP,KXX,1,42
*END
```

The ***CREATE** command takes arguments of the file name, the file extension, and the directory path (in this case, the **macros** directory is specified).

When using ***CREATE**, all parameters used in commands are written to the file (the currently assigned values for the parameter are not substituted).

You cannot use ***CREATE** within a DO loop.

4.2.3.2. Using *CFWRITE

If you wish to create a macro file in which current values are substituted for parameters you can use ***CFWRITE**. Unlike ***CREATE**, the ***CFWRITE** command cannot specify a macro name; you must first specify the macro file with the ***CFOPEN** command. Only those ANSYS commands that are explicitly prefaced with a ***CFWRITE** command are then written to the designated file; all other commands entered in the command input window are executed. As with the ***CREATE** command, ***CFOPEN** can specify a file name, a file extension, and a path. The following example writes a **BLOCK** command to the currently open macro file.

```
*cfwrite,block,,a,,b,,c
```

Note that parameters were used for arguments to the **BLOCK** command. The current value of those parameters (and not the parameter names) are written to the file. So, for this example, the line written to the macro file might be

```
*cfwrite,block,,4,,2.5,,2
```

To close the macro file, issue the ***CFCLOS** command.

Note — While it is possible to create a macro through this method, these commands are most useful as a method for writing ANSYS commands to a file during macro execution.

4.2.3.3. Using /TEE

Issuing **/TEE**,NEW or **/TEE**,APPEND redirects ANSYS commands entered in the command input window to the file designated by the command *at the same time that the commands are being executed*. All commands are executed and redirected until you issue the **/TEE**,END command. If an existing file has the same name as the macro file name you specify with **/TEE**,NEW, the ANSYS program overwrites the existing file. To avoid this, use **/TEE**,APPEND instead.

In addition to the *Label* argument (which can have a value of NEW, APPEND, or END), the **/TEE** command takes arguments of the file name, the file extension, and the directory path.

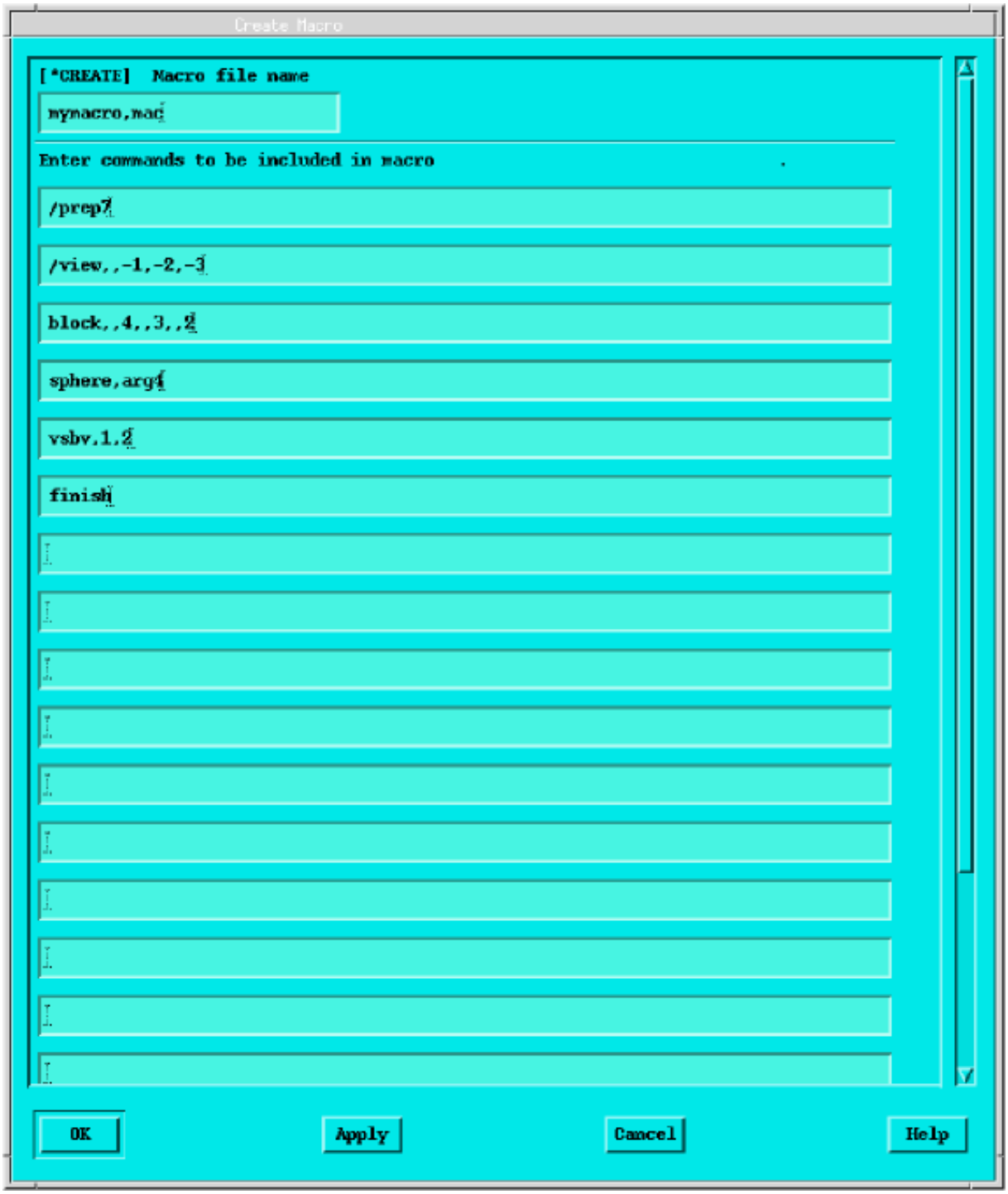
As the commands are executed in the current ANSYS session, all parameter names are resolved to their current values. However, in the file that is created, parameter names are written (the currently assigned values for the parameter are not substituted). If your current parameter values are important, you can save the parameters to a file using the **PARSAV** command.

For an example, see the description of the **/TEE** command in the *ANSYS Commands Reference*.

4.2.3.4. Using Utility Menu> Macro> Create Macro

Choosing this menu item opens an ANSYS dialog box that you can use as a simple editor for creating macros. You cannot open and edit an existing macro with this facility; if you use the name of an existing macro as the arguments for the ***CREATE** field, the existing file will be overwritten.

Figure 4.2 The Create Menu Dialog Box



As with the ***CREATE** command, parameters are not evaluated but are written verbatim into the macro file. Note that you do not make the last line a ***END** command.

4.2.4. Creating Macros with a Text Editor

You can use your favorite text editor to create or edit macro files. Any ASCII editor will work. Moreover, ANSYS macros can have their lines terminated by either UNIX or Windows line ending conventions (carriage-return, line-feed pairs or simply line-feeds) so you can create a macro on one platform and use it on several platforms.

If you use this method to create macros, do not include the ***CREATE** and ***END** commands.

Figure 4.3 A Macro Created in a Text Editor

4.2.5. Using Macro Library Files

As a convenience, ANSYS allows you to place a set of macros in a single file, called a macro library file. You can create these either through the ***CREATE** command or through a text editor. Given that macro libraries tend to be longer than single macros, using a text editor normally provides the best approach.

Macros libraries have no explicit file extension and follow the same file naming conventions as macro files. A macro library file has the following structure:

```
MACRONAME1
.
.
.
/EOF
MACRONAME2
.
.
.
/EOF
MACRONAME3
.
.
.
./EOF
```

For example, the following macro file contains two simple macros:

```
mybloc
/prep7
/view,,-1,-2,-3
block,,4,,3,,2
finish
```

```
/EOF
mysphere
/prep7
/view,,-1,-2,-3
sphere,1
finish
/EOF
```

Note that each macro is prefaced with a macro name (sometimes referred to as a data block name) and ends with a **/EOF** command.

A macro library file can reside anywhere on your system, although for convenience you should place it within the macro search path. Unlike macro files, a macro library file can have any extension up to eight characters.

4.3. Executing Macros and Macro Libraries

You can execute any macro file by issuing the ***USE** command. For example, to execute the macro called **MYMACRO** (no extension) residing somewhere in the macro search path, you would issue

```
*use,mymacro
```

In this case, the macro takes no arguments. If instead the macro was called **MYMACRO.MACRO** and resided in **/myaccount/macros**, you could call it with

```
*use,/myaccount/macros/mymacro.macro
```

Note that the ***USE** command allows you to enter the path and extension along with the file name and that these are not entered as separate arguments.

If a macro *has a* **.mac** *file extension* and resides in the search path, you can execute it as if it were an ANSYS command by simply entering it in the command input window. For example, to call **mymacro.mac** you could simply enter

```
mymacro
```

You can also execute macros with a **.mac** extension through the **Utility Menu> Macro> Execute Macro** menu item.

If the same macro takes arguments (see Section 4.4.1: Passing Arguments to a Macro for more information about passing arguments to macros), then these can be entered on the command line as follows

```
mymacro,4,3,2,1.5
```

or

```
*use,mymacro.mac,4,3,2,1.5
```

The **Utility Menu> Macro> Execute Macro** menu item dialog provides fields for arguments.

Executing macros contained in macro libraries is similar. You must first specify the library file using the ***ULIB** command. For example, to specify that macros are in the **mymacros.mlib** file, which resides in the **/myaccount/macros** directory, you would issue the following command:

```
*ulib,mymacros,mlib,/myaccount/macros/
```

After selecting a macro library, you can execute any macro contained in the library by specifying it through the ***USE** command. As with macros contained in individual files, you can specify arguments as parameters in the ***USE** command.

Note — You cannot use the ***USE** command to access macros not contained in the specified macro library file after issuing the ***ULIB** command.

4.4. Local Variables

APDL provides two sets of specially named scalar parameters which are available for use as local variables. These consist of

- A set of scalar parameters that provide a way of passing command line arguments to the macro.
- A set of scalar parameters that can be used within the macro. These provide a set of local variables that can be used to define values only within that macro.

The following sections discuss both of these variable types in detail.

4.4.1. Passing Arguments to a Macro

There are 19 scalar parameters that you can use to pass arguments from the macro execution command line to the macro. These scalar parameters can be reused with multiple macros; that is, their values are local to each macro. The parameters are named ARG1 through AR19 and they can be used for any of the following items:

- Numbers
- Alphanumeric character strings (up to eight characters enclosed in single quotes)
- Numeric or character parameters
- Parametric expressions

Note — You can pass only the values of parameters ARG1 through AR18 to a macro as arguments with the ***USE** command. If you create a macro that can be used as an ANSYS command (the macro files has a **.mac** extension), you can pass the values of parameters ARG1 through AR19 to the macro.

For example, the following simple macro requires four arguments, *ARG1*, *ARG2*, *ARG3*, and *ARG4*:

```
/prep7
/view,,-1,-2,-3
block,,arg1,,arg2,,arg3
sphere,arg4
vsbv,1,2
finish
```

To execute this macro, a user might enter

```
mymacro,4,3,2.2,1
```

4.4.2. Local Variables Within Macros

Each macro can have up to 79 scalar parameters used as local variables (AR20 through AR99). These parameters are completely local to the macro, and multiple macros can each have their own unique values assigned to these parameters. These parameters are not passed to macros called from macros (nested macros). They are passed to any files processed through a **/INPUT** command or a "do loop" processed within the macro.

4.4.3. Local Variables Outside of Macros

ANSYS also has a similar set of ARG1 through AR99 scalar parameters that are local to an input file, and are not passed to any macros called by that input file. Thus, once a macro finishes and execution returns to an input file, the values of ARG1 through ARG99 revert to whatever values were defined within the input file.

4.5. Controlling Program Flow in APDL

When executing an input file, ANSYS is normally restricted to linear program flow; that is, each statement is executed in the order that it is encountered in the listing. However, APDL provides a rich set of commands that you can use to control program flow.

- Call subroutines (nested macros).
- Branch unconditionally to a specified location with a macro.
- Branch based upon a condition to a specified location within a macro.
- Repeat the execution of a single command, incrementing one or more command parameters.
- Loop through a section of a macro a specified number of times.

The following sections detail each of these program control capabilities. For the exact syntax of the commands, refer to the *ANSYS Commands Reference*.

4.5.1. Nested Macros: Calling Subroutines Within a Macro

APDL allows you to nest macros up to 20 levels deep, providing functionally similar capability to a FORTRAN 77 CALL statement or to a function call. You can pass up to 19 arguments to the macro and, at the conclusion of each nested macro, execution returns to the level that called the macro. For example, the following simply macro library file shows the **MYSTART** macro, which calls the **MYSPHERE** macro to create the sphere.

```
mystart
/prep7
/view,,-1,-2,-3
mysphere,1.2
finish
/eof
mysphere
sphere,arg1
/eof
```

4.5.2. Unconditional Branching: Goto

The simplest branching command, ***GO**, instructs the program to go to a specified label without executing any commands in between. Program flow continues from the specified label. For example

```
*GO,:BRANCH1
---    ! This block of commands is skipped (not executed)
---
:BRANCH1
---
---
```

The label specified by the ***GO** command must start with a colon (:) and must not contain more than eight characters, including the colon. The label can reside anywhere within the same file.

Note — The use of ***GO** is now considered obsolete and is discouraged. See the other branching commands for better methods of controlling program flow.

4.5.3. Conditional Branching: The *IF Command

APDL allows you to execute one of a set of alternative blocks based on the evaluation of a condition. The conditions are evaluated by comparing two numerical values (or parameters that evaluate to numerical values).

The ***IF** command has the following syntax

***IF**, *VAL1*, *Oper*, *VAL2*, *Base*

Where

- *VAL1* is the first numerical value (or numerical parameter) in the comparison.
- *Oper* is the comparison operator.
- *VAL2* is the second numerical value (or numerical parameter) in the comparison.
- *Base* is the action that occurs if the comparison evaluates as true.

APDL offers eight comparison operators, which are discussed in detail in the ***IF** command reference. Briefly these are:

EQ
: Equal (for *VAL1* = *VAL2*).

NE
: Not equal (for *VAL1* ≠ *VAL2*).

LT
: Less than (for *VAL1* < *VAL2*).

GT
: Greater than (for *VAL1* > *VAL2*).

LE
: Less than or equal (for *VAL1* ≤ *VAL2*).

GE
: Greater than or equal (for *VAL1* ≥ *VAL2*).

ABLT
: Absolute values of *VAL1* and *VAL2* before < operation.

ABGT
: Absolute values of *VAL1* and *VAL2* before > operation.

By giving the *Base* argument a value of THEN, the ***IF** command becomes the beginning of an if-then-else construct (similar to the FORTRAN equivalent). The construct consists of

- An ***IF** command, followed by
- One or more optional ***ELSEIF** commands
- An optional ***ELSE** command
- A required ***ENDIF** command, marking the end of the construct.

In its simplest form, the ***IF** command evaluates the comparison and, if true, branches to a label specified in the *Base* argument. This is similar to the "computed goto" in FORTRAN 77. (In combination, a set of such ***IF** commands could function similarly to the CASE statements in other programming languages.) Take care not to branch to a label within an if-then-else construct or do-loop. If a batch input stream hits an end-of-file during a false ***IF** condition, the ANSYS run will not terminate normally. You will need to terminate it externally (use either the UNIX “kill” function or the Windows task manager).

By setting the *Base* argument to a value of STOP, you can exit from ANSYS based on a particular condition.

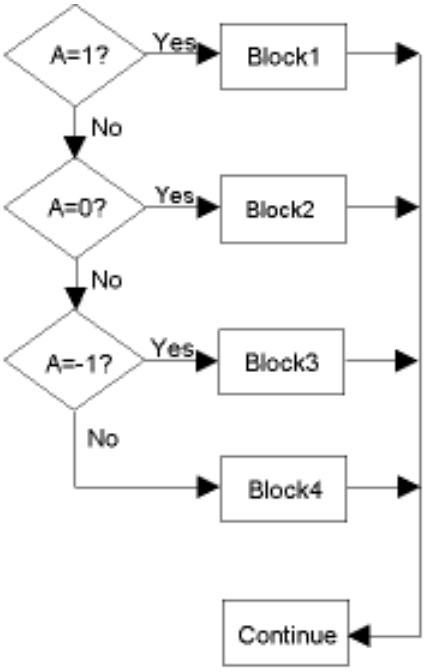
An if-then-else construct simply evaluates a condition and executes the following block or jumps to the next statement following the ***ENDIF** command (shown with the "Continue" comment).

```
*IF,A,EQ,1,THEN
    !  Block1
    .
    .
*ENDIF
! Continue
```

The following example shows a more complex structure. Note that only one block can be executed. If no comparison evaluates to true, the block following the ***ELSE** command is executed.

Figure 4.4 A Sample If-Then-Else Construct

```
*IF,A,EQ,1,THEN
    ! Block1
    .
    .
    *ELSEIF,A,EQ,0
     ! Block2
     .
     .
     *ELSEIF,A,EQ,-1
     ! Block3
     .
     .
     *ELSE
     ! Block4
     .
     .
*ENDIF
! Continue
```



Note — You can issue a **/CLEAR** command within an if-then-else construct. The **/CLEAR** command does *not* clear the ***IF** stack and the number of ***IF** levels is retained. An ***ENDIF** is necessary to close any branching logic. Also, keep in mind that the **/CLEAR** command deletes all parameters, including any that are used in your branching commands. You can avoid any problems that might arise from the deletion of parameters by issuing a **PARSAV** command before the **/CLEAR** command, and then following the **/CLEAR** command with a **PARRES** command.

4.5.4. Repeating a Command

The simplest looping capability, the ***REPEAT** command, allows you to execute the directly preceding command a specified number of times, incrementing any field in that command by a constant value. In the example

```
E,1,2
*REPEAT,5,0,1
```

the **E** command generates one element between nodes 1 and 2 and the following ***REPEAT** command specifies that **E** executes a total of five times (including the original **E** command), incrementing the second node number by one for each additional execution. The result is five total elements with node connectivities 1-2, 1-3, 1-4, 1-5, and 1-6.

Note — Most commands that begin with a slash (/) or an asterisk (*), as well as macros executed as "unknown commands," cannot be repeated. However, graphics commands that begin with a slash can be repeated. Also, avoid using the ***REPEAT** command with interactive commands, such as those that require picking or those that require a user response.

4.5.5. Looping: Do-Loops

A do-loop allows you to loop through a series of commands a specified number of times. The ***DO** and ***ENDDO** commands mark the beginning and ending points for the loop. ***DO** command has the following syntax:

The following example do-loop edits five load step files (numbered 1 through 5) and makes the same changes in each file.

```
*DO,I,1,5       ! For I = 1 to 5:
LSREAD,I        ! Read load step file I
OUTPR,ALL,NONE  ! Change output controls
ERESX,NO
LSWRITE,I       ! Rewrite load step file I
*ENDDO
```

You can add your own loop controls by using the ***IF**, ***EXIT**, or ***CYCLE** commands.

Keep the following guidelines in mind when constructing do-loops.

- Do not branch out of a do-loop with a *:Label* on the ***IF** or ***GO** commands.
- Avoid using a *:Label* to branch to a different line within a do-loop. Use if-then-else-endif instead.
- Output from commands within a do-loop is automatically suppressed after the first loop. Use **/GOPR** or **/GO** (no response line) within the do-loop if you need to see output for all loops.
- Take care if you include a **/CLEAR** command within a do-loop. The **/CLEAR** command does not clear the do-loop stack, but it does clear all parameters including the loop parameter in the ***DO** statement itself. You can avoid the problem of having an undefined looping value by issuing a **PARSAV** command before the **/CLEAR** command, and then following the **/CLEAR** command with a **PARRES** command.

4.5.6. Implied (colon) Do Loops

You can also use the implied (colon) convention for do loops. Using this convention is typically faster because the looping is done in memory. The correct syntax is:

```
(x:y:z)
```

with z defaulting to 1 if not specified. For example:

```
n,(1:6),(2:12:2)
```

will perform the same steps as:

```
n,1,2
n,2,4
n,3,6
.
.
.
n,6,12
```

When using the implied (colon) do loops, be aware that the shortest expression controls execution. For example,

```
n,(1:7),(2:12:2)
```

would behave identically to the example above.

Additional numeric fields that do not have the colon (:) will be taken as a constant value.

Also, non-integer numbers will function normally. However, if non-integer numbers are applied to a command that requires integers, then the non-integer will be rounded off following normal mathematical conventions.

This looping convention can be used only for fields requiring a numeric entry. A text entry field will process (x:y:z) as a literal value.

4.5.7. Additional Looping: Do-While

You can also perform looping functions that will repeat indefinitely until an external parameter changes. The ***DOWHILE** command has the following syntax:

***DOWHILE**,*Parm*

The loop repeats as long as the parameter *Parm* is TRUE. If *Parm* becomes false (less than or equal to 0.0), the loop terminates. The ***CYCLE** and ***EXIT** commands can be used within a ***DOWHILE** loop.

4.6. Control Functions Quick Reference

The table below describes APDL commands that perform control functions within macros.

Most of the important information about these commands appears here, but you may want to look at the complete command descriptions in the *ANSYS Commands Reference*.

APDL Command	Action It Takes	Usage Tips
***DO**	Defines the start of a "do" loop. The commands following the ***DO** command execute (up to the ***ENDDO** command) repeatedly until some loop control is satisfied.	• You also can control looping via the ***IF** command. • ANSYS allows up to 20 levels of nested “do” loops, although “do” loops that include **/INPUT**, ***USE**, or an “unknown” command macro support fewer nesting levels because they do internal file switching. • ***DO**, ***ENDDO**, ***CYCLE**, and ***EXIT** commands in a “do” loop must all read from the same file or the keyboard. • *Do not include picking operations in a “do” loop.* • Be careful if you include a **/CLEAR** command within a do-loop. The **/CLEAR** command does not clear the do-loop stack, but it does clear all parameters including the loop parameter in the ***DO** statement itself. You can avoid the problem of having an undefined looping value by issuing a **PARSAV** command before the **/CLEAR** command, and then following the **/CLEAR** command with a **PARRES** command.
***ENDDO**	Ends a "do" loop and starts the looping action.	You must use one ***ENDDO** command for each nested "do" loop. The ***ENDDO** and ***DO** commands for a loop must be on the same file.
***CYCLE**	When executing a "do" loop, ANSYS bypasses all commands between the ***CYCLE** and ***ENDDO** commands, then (if applicable) initiates the next loop.	You can use the cycle option conditionally (via the ***IF** command). The ***CYCLE** command must appear on the same file as the ***DO** command and must appear before the ***ENDDO** command.

APDL Command	Action It Takes	Usage Tips
***EXIT**	Exits from a "do" loop.	The command following the ***ENDDO** command executes next. The ***EXIT** and ***DO** commands for a loop must be on the same file. You can use the exit option conditionally (via the ***IF** command).
***IF**	Causes commands to be read conditionally.	• You can have up to 10 nested levels of ***IF** blocks. • You cannot jump into, out of, or within a "do" loop or an if-then-else construct to a *:label* line, and jumping to a *:label* line is not allowed with keyboard entry. • You can issue a **/CLEAR** command within an if-then-else construct. The **/CLEAR** command does *not* clear the ***IF** stack and the number of ***IF** levels is retained. An ***ENDIF** is necessary to close any branching logic. • The **/CLEAR** command deletes all parameters, including any that are used in your branching commands. You can avoid any problems that might arise from the deletion of parameters by issuing a **PARSAV** command before the **/CLEAR** command, and then following the **/CLEAR** command with a **PARRES** command.
***ENDIF**	Terminates an if-then-else construct. (See the ***IF** discussion for details.)	The ***IF** and ***ENDIF** commands must appear in the same file.
***ELSE**	Creates a final, optional block separator within an if-then-else construct. (See the ***IF** discussion for details.)	The ***ELSE** and ***IF** commands must appear in the same file.
***ELSEIF**	Creates an optional, intermediate block separator within an if-then-else construct.	If *Oper* = EQ or NE, *VAL1* and *VAL2* can also be character strings (enclosed in quotes) or parameters. The ***IF** and ***ELSEIF** commands must be on the same file.

4.7. Using the _STATUS and _RETURN Parameters in Macros

The ANSYS program generates two parameters, _STATUS and _RETURN, that you can also use in your macros. For example, you might use the _STATUS or _RETURN value in an "if-then-else" construct to have the macro take some action based on the outcome of executing an ANSYS command or function.

Solid modeling functions generate the _RETURN parameter, which contains the result of executing the function. The following table defines the _RETURN values for the various solid modeling functions:

Table 4.1 _RETURN Values

Command	Function	_RETURN Value
Keypoints		
K	Defines a keypoint	keypoint number
KL	Keypoint on a line	Keypoint number
KNODE	Keypoint at node	Keypoint number
KBETW	Keypoint between two keypoints	KP number

Command	Function	_RETURN Value
KCENTER	Keypoint at center	KP number
Lines		
BSPLIN	Generate spline	Line number
CIRCLE	Generate circular arc lines	First line number
L	Line between two keypoints	Line number
L2ANG	Line at angle with two lines	Line number
LANG	Line tangent to two lines	Line number
LARC	Defines a circular arc	Line number
LAREA	Line between two keypoints	Line number
LCOMB	Combine two lines into one	Line number
LDIV	Divide line into two or more lines	First keypoint number
LDRAG	Line by keypoint sweep	First line number
LFILLT	Fillet line between two liens	Fillet line number
LROTAT	Arc by keypoint rotation	First line number
LSTR	Straight line	Line number
LTAN	Line at end and tangent	Line number
SPLINE	Segmented spline	First line number
Areas		
A	Area connecting keypoints	Area number
ACCAT	Concatenate two or more areas	Area number
ADRAG	Drag lines along path	First area number
AFILLT	Fillet at intersection of two areas	Fillet area number
AL	Area bounded by lines	Area number
ALPFILL	All loops	Area number
AOFFST	Area offset from given area	Area number
AROTAT	Rotate lines around axis	First area number
ASKIN	Skin surface through guiding lines	First area number
ASUB	Area using shape of existing area	Area number
Volumes		
V	Volume through keypoints	Volume number
VA	Volume bounded through areas	Volume number
VDRAG	Drag area pattern to create volume	First volume number
VEXT	Volume by extruding areas	First volume number
VOFFST	Volume offset from given area	Volume number
VROTAT	Volume by rotating areas	First volume number

Executing an ANSYS command, whether in a macro or elsewhere, generates the parameter _STATUS. This parameter reflects the error status of that command:

- 0 for no error
- 1 for a note

- 2 for a warning
- 3 for an error

4.8. Using Macros with Components and Assemblies

To make large models easier to manage, you may want to divide a model into discrete components based on different types of entities: nodes, elements, keypoints, lines, areas, or volumes. Each component can contain only one type of entity. Doing this enables you to perform tasks such as applying loads or producing graphics displays conveniently and separately on different portions of the model.

You can also create assemblies, which are groups that combine two or more components or even multiple assemblies. You can nest assemblies up to five levels deep. For example, you could build an assembly named motor from components called STATOR, PERMMAG, ROTOR, and WINDINGS.

The table below describes some of the commands you can issue to build components and assemblies. For more detailed discussions of these commands, see the *ANSYS Commands Reference*. For further information on components and assemblies, see *Selecting and Components* in the *ANSYS Basic Analysis Guide*.

CM	Groups geometry items into a component
CMDELE	Deletes a component or assembly.
CMEDIT	Edits an existing component or assembly. ANSYS updates assemblies automatically to reflect deletions of lower-level or assemblies.
CMGRP	Groups components and assemblies into one assembly. Once defined, an assembly can be listed, deleted, selected, or unselected using the same commands as for components.
CMLIST	Lists the entities contained in a component or assembly.
CMSEL	Selects a subset of components and assemblies.

4.9. Reviewing Example Macros

Following are two example macros. The example macro below, called **offset.mac**, offsets selected nodes in the PREP7 preprocessor. This macro is for demonstration purposes only because the **NGEN** command provides a more convenient method.

```
!   Macro to offset selected nodes in PREP7
!   The below file is saved as:  offset.mac (must be lowercase)
!   Usage:  offset,dx,dy,dz

/nop            ! suppress printout for this macro

*get,nnode,node,,num,max   ! get number of nodes

*dim,x,,nnode       ! set up arrays for node locations
*dim,y,,nnode
*dim,z,,nnode

*dim,sel,,nnode     ! set up array for select vector

*vget,x(1),node,1,loc,x   ! get coordinates
*vget,y(1),node,1,loc,y
*vget,z(1),node,1,loc,z

*vget,sel(1),node,1,nsel  ! get selected set

*voper,x(1),x(1),add,arg1 ! offset locations
*voper,y(1),y(1),add,arg2
*voper,z(1),z(1),add,arg3

! *do,i,1,nnode           ! store new positions
!  *if,sel(i),gt,0,then   ! this form takes 98 sec for 100,000 nodes
```

```
!    n,i,x(i),y(i),z(i)
!  *endif
! *enddo

*vmask,sel(1)         ! takes 3 seconds for 100,000 nodes
n,(1:NNODE),x(1:NNODE),y(1:NNODE),z(1:NNODE)

x(1) =       ! delete parameters (cleanup)
y(1) =
z(1) =
sel(1) =
i=
nnode=

/go          ! resume printout
```

The following example macro, called **bilinear.mac**, evaluates two bilinear materials. This is a useful macro that can be run after solving a static analysis. Material 1 is the tension properties, and Material 2 is the compression properties. ARG1 is the number of iterations (default is 2).

```
/nop
_niter = arg1                     ! set number of iterations
*if,_niter,lt,2,then
   _Niter = 2
*endif
*do,iter,1,_niter                 ! loop on number of iterations
/post1
set,1,1
ar11,=elmiqr(0,14)                ! call elmiqr function to get no. of elements
*dim,_s1,,ar11                    ! array for element s1
*dim,_s3,,ar11                    ! array for element s3
etable,sigmax,s,1                 ! s1 is in element table sigmax
etable,sigmin,s,3                 ! s3 is in element table sigmin
*vget,_s1(1),elem,1,etab,sigmax   ! get element maximum stress in s1
*vget,_s3(1),elem,1,etab,sigmin   ! get element minimum stress in s3
*dim,_mask,,ar11                  ! array for mask vector
*voper,_mask(1),_s1(1),lt,0       ! true if max. stress &lt; 0
*vcum,1                           ! accumulate compression elements
*vabs,0,1                         ! absolute value of s3
*voper,_mask(1),_s3(1),gt,_s1(1)  ! true if abs(minstr) > maxstr
finish

/prep7                            ! go to prep7 for element material mods
mat,1                             ! set all materials to tension properties
emod,all

*vput,_mask(1),elem,1,esel        ! select compression elements
mat,2                             ! change selected elements to compression
emod,all

call                              ! select all elements
finish

_s1(1)=                           ! clean up all vectors (set to zero)
_s3(1)=
_mask(1)=

/solve                            ! rerun the analysis
solve
finish

*enddo                            ! end of iterations

_niter=                           ! clean up iteration counters
_iter=
/gop
```

Chapter 5: Interfacing with the GUI

Within an ANSYS macro, you have several ways to access components of the ANSYS graphical user interface (GUI):

- You can modify and update the ANSYS toolbar (this is discussed in detail in Section 2.1: Adding Commands to the Toolbar).
- You can issue the ***ASK** command to prompt a user to enter a single parameter value.
- You can create a dialog box to prompt a user to enter multiple parameter values.
- You can issue the ***MSG** command to have the macro write an output message.
- You can have the macro update or remove a status bar.
- You can allow the user to select entities through graphical picking from within a macro.
- You can call any dialog box.

5.1. Prompting Users for a Single Parameter Value

By including the ***ASK** command within a macro, you can have the macro prompt a user to type in a parameter value.

The format for the ***ASK** command is

***ASK**,*Par*,*Query*,*DVAL*

Where

- *Par* is an alphanumeric name that identifies the scalar parameter used to store the user input.
- *Query* is the text string that ANSYS displays to prompt the user. This string can contain up to 54 characters. *Don't use characters that have special meanings, such as "$" or "!".*
- *DVAL* is the default value given the parameter if a user issues a blank response. This value can be either a one-to-eight character string (enclosed in single quotes) or a number. If you assign no default, a blank user response deletes the parameter.

The ***ASK** command prints the *Query* text on the screen and waits for a response. It reads the response from the keyboard except when ANSYS runs in batch mode. (In that case, the response or responses must be the next-read input line or lines.) The response can be a number, a one-to-eight character string enclosed in single quotes, a numeric or character parameter, or an expression that evaluates to a number. ANSYS then sets the value of *Par* to the read-in response. The following example displays the dialog box shown below, then sets the parameter PARM1 to the value the user enters.

```
*ask,parm1,'username (enclose the username in single quotes)'
```

Figure 5.1 An Example *ASK Dialog Box

When you issue ***ASK** within a macro, ANSYS writes the user's response to **File.LOG** on the line following the macro name.

5.2. Prompting Users With a Dialog Box

The **MULTIPRO** command constructs a simple, multiple-prompt dialog box that can contain up to 10 parameter prompts. The command allows you to use a set of UIDL ***CSET** commands to create the prompts as well as specify a default value for each prompt. Be aware that macros using **MULTIPRO** cannot be called from UIDL. You cannot use **MULTIPRO** within a DO loop.

The **MULTIPRO** command must be used in conjunction with:

- Between one and ten ***CSET** command prompts
- Up to two special ***CSET** commands that provide a two line area for user instructions.

The command has the following syntax:

```
MULTIPRO,'start',Prompt_Num
*CSET,Strt_Loc,End_Loc,Param_Name,'Prompt_String',Def_Value
MULTIPRO,'end'
```

Where

'start'
: A literal string that, when encountered as the first argument, marks the beginning of the **MULTIPRO** construct. The literal must be enclosed in single quotes.

Prompt_Num
: Required only if *Def_Value* is omitted from at least one ***CSET** command or if *Def_Value* is set to 0. The *Prompt_Num* value is an integer equal to the number of following ***CSET** prompts.

Strt_Loc,*End_Loc*
: The initial value for *Strt_Loc* for the first ***CSET** command is 1, and the value for *End_Loc* is *Strt_Loc*+2 (3 for the first ***CSET** command). The value of each subsequent *Strt_Loc* is the previous *End_Loc*+1.

Param_Name
: The name of the parameter that will hold either the value specified by the user or, if the user supplies no value, the value of *Def_Value* .

''*Prompt_String*''
: A string, which can contain up to 32 characters, which can be used to describe the parameter. This string must be enclosed in single quotes.

Def_Value
Default value used if no value specified by user. Default value can be a numeric expression including APDL numeric parameters. Character expressions are not allowed.

'end'
A literal string, used as the first argument for the closing **MULTIPRO** command.

The following is a typical example of the **MULTIPRO** command.

```
multipro,'start',3
    *cset,1,3,beamW,'Enter the overall beam width',12.5
    *cset,4,6,beamH,'Enter the beam height',23.345
    *cset,7,9,beamL,'Enter the beam length',50.0
multipro,'end'
```

Up to two optional ***CSET** commands can be added to the construct that can provide two 64 character strings. You can use these to provide instructions to the user. The syntax for these specialized ***CSET** commands is

CSET**,61,62,*'Help_String'*,*'Help_String'*CSET**,63,64,'*Help_String*',*'Help_String'*

Where

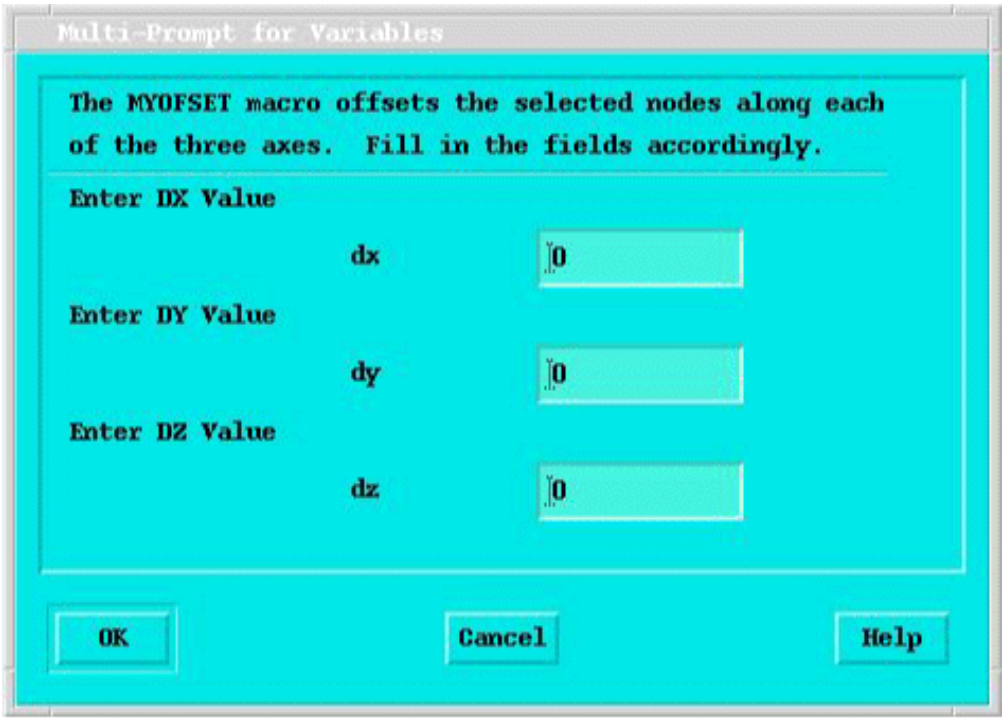
'Help_String'
A string which can contain up to 32 characters. If you need more than 32 characters, you can use a second *Help_String* argument.

The following is an example of a **MULTIPRO** construct using the optional help lines. Note that two *Help_String* arguments are used to overcome the 32 character limit.

```
multipro,'start',3
    *cset,1,3,dx,'Enter DX Value',0.0
    *cset,4,6,dy,'Enter DY Value',0.0
    *cset,7,9,dz,'Enter DZ Value',0.0
    *cset,61,62,'The MYOFSET macro offsets the',' selected nodes along each'
    *cset,63,64,'of the three axes.  Fill in the ',' fields accordingly.'
multipro,'end'
```

The above construct creates the following multiple-prompt dialog box.

Figure 5.2 A Typical Multiple-Prompt Dialog Box



You can check the status of the buttons by testing the value of the _BUTTON parameter. The following lists the button status values:

- _BUTTON = 0 indicates that the OK button was pressed.
- _BUTTON = 1 indicates that the Cancel button was pressed.

At present, the Help button is not functional.

5.3. Using Macros to Display Your Own Messages

By issuing the ***MSG** command within a macro, you can display custom output messages via the ANSYS message subroutine. The command has the following format:

```
*MSG,Lab,VAL1,VAL2,VAL3,VAL4,VAL5,VAL6,VAL7,VAL8
```

Where *Lab* is one of the following labels for output and termination control:

INFO	Writes the message with no heading (default).
NOTE	Writes the message with a "NOTE" heading.
WARN	Writes the message with a "WARNING" heading, and also writes it to the errors file, **Jobname.ERR**.
ERROR	Writes the message with an "ERROR" heading and also writes it to the errors file, **Jobname.ERR**. If this is an ANSYS batch run, this label also terminates the run at the earliest "clean exit" point.
FATAL	Writes the message with a "FATAL ERROR" heading and also writes it to the errors file, **Jobname.ERR**. This label also terminates the ANSYS run immediately.
UI	Writes the message with a "NOTE" heading and displays it in the message dialog box.

VAL1 through VAL8 are numeric or alphanumeric character values to be included in the message. Values can be the results of evaluating parameters. All numeric values are assumed to be double precision.

You must specify the message format immediately after the ***MSG** command. The message format can contain up to 80 characters, consisting of text strings and predefined "data descriptors" between the strings where numeric or alphanumeric character data are to be inserted. These data descriptors are:

- %i, for integer data. The FORTRAN nearest integer (NINT) function is used to form integers for the %I specifier.`
- %g, for double precision data
- %c, for alphanumeric character data
- %/, for a line break

The corresponding FORTRAN data descriptors for the first three descriptors are I9, 1PG16.9, and A8 respectively. *A blank must precede each descriptor.* You also must supply one data descriptor for each specified value (eight maximum), in the order of the specified values.

Don't begin ***MSG** format lines with ***IF**, ***ENDIF**, ***ELSE**, or ***ELSEIF**. If the last non-blank character of the message format is an ampersand (&), the ANSYS program reads a second line as a continuation of the format. You can use up to 10 lines (including the first) to specify the format information.

Consecutive blanks are condensed into one blank upon output, and a period is appended. The output produced can be up to 10 lines of 72 characters each (using the $/ descriptor).

The example below shows you an example of using ***MSG** that prints a message with two integer values and one real value:

```
*MSG, INFO, 4Inner4 ,25,1.2,148
Radius ( %C) = %I, Thick = %G, Length = %I
```

The resulting output message is as follows:

```
Radius (Inner) = 25, Thick = 1.2, Length = 148
```

Here is an example illustrating multiline displays in GUI message windows:

```
*MSG,UI,Vcoilrms,THTAv,Icoilrms,THTAi,Papprnt,Pelec,PF,indctnc
Coil RMS voltage, RMS current, apparent pwr, actual pwr, pwr factor: %/&
Vcoil = %G V (electrical angle = %G DEG) %/&
Icoil = %G A (electrical angle = %G DEG) %/&
APPARENT POWER = %G W %/&
ACTUAL POWER = %G W %/&
Power factor: %G %/&
Inductance = %G %/&
VALUES ARE FOR ENTIRE COIL (NOT JUST THE MODELED SECTOR)
```

Note — The command **/UIS**,MSGPOP controls which messages a message dialog box displays when the GUI is active. See the *ANSYS Commands Reference* for more information about this command.

5.4. Creating and Maintaining a Status Bar from a Macro

Within macros, you can insert commands to define an ANSYS dialog box containing a status bar displaying the progress of an operation, a STOP button you can click on to stop the operation, or both.

To define a status dialog box, issue the following command:

```
*ABSET,Title40,Item
```

- *Title40* is the text string that appears in the dialog box with the status bar. The string can contain a maximum of 40 characters.

- *Item* is one of the following values:

BAR	Displays the status bar with no STOP button
KILL	Displays a STOP button with no status bar
BOTH	Displays both the status bar and STOP button

To update the status bar, issue the command ***ABCHECK**,*Percent*,*NewTitle*.

- *Percent* is an integer between 0 and 100. It gives the position of the status bar.
- *NewTitle* is a 40-character string that contains progress information. If you specify a string for *NewTitle*, it replaces the string supplied in *Title40*.

If you specify KILL or BOTH, your macro should check the _RETURN parameter after each execution of ***ABCHECK** to see if the user has pressed the STOP button, then take the appropriate action.

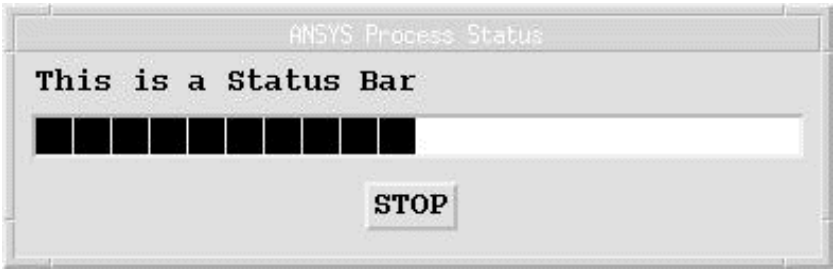
To remove the status bar from the ANSYS GUI, issue the ***ABFINI** command.

The following example macro illustrates the status bar (complete with bar and STOP button) in use. The status dialog box that is produced is shown in the following figure. Note that the macro checks the status of the _RETURN parameter and, if the STOP button is pressed, posts the "We are stopped....." message.

```
fini
/clear,nost
/prep7
n,1,1
n,1000,1000
fill
*abset,'This is a Status Bar',BOTH
myparam = 0
*do,i,1,20
   j = 5*i
   *abcheck,j
   *if,_return,gt,0,then
     myparam = 1
   *endif
   *if,myparam,gt,0,exit
   /ang,,j
   nplot,1
   *if,_return,gt,0,then
     myparam = 1
   *endif
   *if,myparam,gt,0,exit
   nlist,all
   *if,_return,gt,0,then
     myparam = 1
   *endif
   *if,myparam,gt,0,exit
*enddo
*if,myparam,gt,0,then
*msg,ui
We are stopped.........
*endif
*abfinish
fini
```

Note — Do not call ***ABCHECK** more than about 20 times in a loop.

Figure 5.3 A Typical Status Dialog Box



5.5. Picking within Macros

If you're running the ANSYS program interactively, you can call a GUI picking menu from within a macro. To do so, simply include a picking command in the macro. Many ANSYS commands (such as **K**,,P) accept the input "P" to enable graphical picking. When ANSYS encounters such a command, it displays the appropriate picking dialog and then continues macro execution when the user clicks OK or Cancel.

Keep in mind that picking commands are not available in all ANSYS processors, and that you must first switch to an appropriate processor before calling the command.

Note — If a macro includes GUI functions, the **/PMACRO** command should be the first command in that macro. This command causes the macro contents to be written to the session log file. This is important, because if you omit the **/PMACRO** command, ANSYS can't read the session log file to reproduce the ANSYS session.

5.6. Calling Dialog Boxes From a Macro

When the ANSYS program encounters a dialog box UIDL function name (such as Fnc_UIMP_Iso), it displays the appropriate dialog box. Thus, you can launch any ANSYS dialog box by making its function name a separate line in the macro file. When you dismiss that dialog box, the program continues processing the macro starting with the next line after the function call.

Keep in mind that many dialog boxes have a number of dependencies, including that the appropriate ANSYS processor is active and that certain required preexisting conditions are met. For example, launching a dialog box to select nodes first supposes that nodes exist, if no nodes exist the macro will fail when the user clicks OK or Apply.

Note — If a macro includes GUI functions, the **/PMACRO** command should be the first command in that macro. This command causes the macro contents to be written to the session log file. This is important, because if you omit the **/PMACRO** command, ANSYS can't read the session log file to reproduce the ANSYS session.

Chapter 6: Encrypting Macros

ANSYS provides the ability to encrypt macro files so that the source is not "human-readable." Encrypted macros require an encryption key to run. You can either place the encryption key explicitly (in readable ASCII) in the macro or you can set it in ANSYS as a global encryption key.

6.1. Preparing a Macro for Encryption

Before encrypting a macro, you first create and debug the macro as usual. *When you create an encrypted macro, you are responsible for keeping the original source file. You cannot recreate the source file from an encrypted macro.* You then add an **/ENCRYPT** command as the first line and last of the macro. The **/ENCRYPT** command for the first line of the macro has the following syntax:

/ENCRYPT,*Encryption_key*,*File_name*,*File_ext*,*Directory_Path/*

Where

- *Encryption_key* is an eight-character password.
- *File_name* is the name of the encrypted macro filename.
- *File_ext* is an optional file extension for the encrypted macro file. If you want users to execute the macro as an "unknown" command, you should use the **.mac** extension.
- *Directory_Path/* is the optional directory path that can contain up to 60 characters; you only need this argument if you do not want to write the encrypted macro file to your "home" directory.

Note the placement of the **/ENCRYPT** commands at the top and bottom of the listing in the following example:

```
/encrypt,mypasswd,myenfile,mac,macros/
/nopr
/prep7
/view,,-1,-2,-3
block,,arg1,,arg2,,arg3
sphere,arg4
vsbv,1,2
/gopr
finish
/encrypt
```

The **/ENCRYPT** command at the top of the macro instructs ANSYS to encrypt the file and use the string "mypasswd" as the encryption key. It will create an encrypted macro file called **myenfile.mac** and place it in the **/macros** subdirectory of the home directory. The **/ENCRYPT** command at the bottom instructs ANSYS to stop the encryption process and write the encrypted macro to the specified file.

> *Note* — The encrypted macro uses a **/NOPR** command as its second line to turn off echoing of ANSYS commands to the session log file. This is important if you wish to prevent users from reading the contents of the macro from the session log. It's a good practice to reactivate the session log by issuing the **/GOPR** command as the last command in the macro before the ending **/ENCRYPT** command.

6.2. Creating an Encrypted Macro

After putting the **/ENCRYPT** commands at the top and bottom of the macro, you can proceed to create the encrypted version of the macro. To do this, simply execute the macro through ANSYS. ANSYS will create the encrypted version with the name and location you specified through the **/ENCRYPT** command at the top of the macro. The result should look something like this

```
/DECRYPT,mypasswd
013^Z,^%
02x^0Se|Lv(yT.6>?
03J3]Q_LuXd3-6=m+*f$k]?eB
04:^VY7S#S>c>
05daV;u(yY
06T]3WjZ
/DECRYPT
```

Note that the individual commands within the macro are now encrypted, and that the encrypted material is enclosed by **/DECRYPT** commands. The encryption key is the argument to the first **/DECRYPT** command.

6.3. Running an Encrypted Macro

You can run an encrypted macro just as you would any other macro; place the encrypted macro within the macro search path. If you would prefer to run the encrypted macro without having the encryption key resident in the macro file, you can define the key as a "global encryption key" within ANSYS. To do this you must first replace the encryption key argument in the **/DECRYPT** command with the parameter PASSWORD. Thus, the first line of the encrypted macro becomes:

```
/DECRYPT,PASSWORD
```

Before executing the macro within ANSYS, issue the following command through the ANSYS Input command line:

```
/DECRYPT,PASSWORD,Encryption_Key
```

Where *Encryption_Key* is the encryption key used to encrypt the file. You can now execute the encrypted password. To delete the current global encryption key, issue the following ANSYS command:

```
/DECRYPT,PASSWORD,OFF
```

APDL Commands Reference

***ABBR**, *Abbr*, *String*
Defines an abbreviation.

APDL: Abbreviations

MP ME ST DY <> PR EM <> FL PP ED

Abbr
The abbreviation (up to 8 alphanumeric characters) used to represent the string *String*. If *Abbr* is the same as an existing ANSYS command, the abbreviation overrides. Avoid using an *Abbr* which is the same as an ANSYS command.

String
String of characters (60 maximum) represented by *Abbr*. Cannot include a $ or any of the commands **C*****, **/COM**, **/GOPR**, **/NOPR**, **/QUIT**, **/UI**, or ***END**. Parameter names and commands of the ***DO** and Use the ***IF** groups may not be abbreviated. If *String* is blank, the abbreviation is deleted. To abbreviate multiple commands, create an "unknown command" macro or define *String* to execute a macro file [***USE**] containing the desired commands.

Notes

Once the abbreviation *Abbr* is defined, you can issue it at the beginning of a command line and follow it with a blank (or with a comma and appended data), and the program will substitute the string *String* for *Abbr* as the line is executed. Up to 100 abbreviations may exist at any time and are available throughout the program. Abbreviations may be redefined or deleted at any time.

Use ***STATUS** to display the current list of abbreviations. For abbreviations repeated with ***REPEAT**, substitution occurs before the repeat increments are applied. There are a number of abbreviations that are predefined by the program (these can be deleted by using the blank *String* option described above). Note that *String* will be written to the **File.LOG**.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Edit Abbreviations
Utility Menu>MenuCtrls>Edit Toolbar

ABBRES, *Lab*, *Fname*, *Ext*, --
Reads abbreviations from a coded file.

APDL: Abbreviations

MP ME ST DY <> PR EM <> FL PP ED

Lab
Label that specifies the read operation:

NEW --
Replace current abbreviation set with these abbreviations (default).

CHANGE --
Extend current abbreviation set with these abbreviations, replacing any of the same name that already exist.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

The file name defaults to **Jobname**.

Ext
Filename extension (8 character maximum).

The extension defaults to ABBR if *Fname* is blank.

--
Unused field

Notes

The abbreviation file may have been written with the **ABBSAV** command. Do not issue **ABBRES**,NEW while inside an executing abbreviation. Doing so will cause all data for the executing abbreviation to be deleted.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Restore Abbr
Utility Menu>MenuCtrls>Restore Toolbar

ABBSAV, *Lab*, *Fname*, *Ext*, --
Writes the current abbreviation set to a coded file.

APDL: Abbreviations
MP ME ST DY <> PR EM <> FL PP ED

Lab
Label that specifies the write operation:

ALL --
Write all abbreviations (default).

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

The file name defaults to **Jobname**.

Ext
Filename extension (8 character maximum).

The extension defaults to ABBR if *Fname* is blank.

- -
Unused field

Notes

Existing abbreviations on this file, if any, will be overwritten. The abbreviation file may be read with the **ABBRES** command.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Save Abbr
Utility Menu>MenuCtrls>Save Toolbar

---

*AFUN, *Lab*

Specifies units for angular functions in parameter expressions.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Lab
Specifies the units to be used:

RAD --
Use radians for input and output of parameter angular functions (default).

DEG --
Use degrees for input and output of parameter angular functions.

STAT --
Show current setting (DEG or RAD) for this command.

Command Default

Use radians for input or output of parameter angular functions.

Notes

Only the SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2, ANGLEK, and ANGLEN functions [***SET**,***VFUN**] are affected by this command.

Menu Paths

Utility Menu>Parameters>Angular Units

***ASK**, *Par*, *Query*, *DVAL*

Prompts the user to input a parameter value.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par

An alphanumeric name used to identify the scalar parameter. See ***SET** for name restrictions.

Query

Text string to be displayed on the next line as the query (32 characters maximum). Characters having special meaning (such as $! ,) should not be included.

DVAL

Default value assigned to the parameter if the user issues a blank response. May be a number or character string (up to 8 characters enclosed in single quotes). If a default is not assigned, a blank response will delete the parameter.

Notes

Intended primarily for use in macros, the command prints the query (after the word ENTER) on the next line and waits for a response. The response is read from the keyboard, except in batch mode [**/BATCH**], when the response(s) must be the next-read input line(s). The response may be a number, a character string (up to 8 characters enclosed in single quotes), a parameter (numeric or character) or an expression that evaluates to a number. The scalar parameter is then set to the response value. For example, ***ASK**,NN,PARAMETER NN will set NN to the value entered on the next line (after the prompt ENTER PARAMETER NN).

The ***ASK** command is not written to **File.LOG**, but the responses are written there as follows: If ***ASK** is contained in a macro, the response(s) (only) is written to **File.LOG** on the line(s) following the macro name. If not contained in a macro, the response is written to **File.LOG** as a parameter assignment (i.e., *Par* = "user-response").

If used within a do-loop that is executed interactively, ***ASK** should be contained in a macro. If not contained in a macro, ***ASK** will still query the user as intended, but the resulting log file will *not* reproduce the effects of the original run.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*CFCLOS

Closes the "command" file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Notes

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*CFOPEN, *Fname*, *Ext*, --, *Loc*

Opens a "command" file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Fname

File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

The file name defaults to **Jobname**.

Ext

Filename extension (8 character maximum).

The extension defaults to CMD if *Fname* is blank.

--

Unused field

Loc

Determines whether existing file will be overwritten or appended:

(blank) --

The existing file will be overwritten.

APPEND --

The file will be appended to the existing file.

Notes

Data processed with the ***VWRITE** command will also be written to this file if the file is open when the ***VWRITE** command is issued.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

***CFWRITE**, *Command*

Writes an ANSYS command (or similar string) to a "command" file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Command

Command or string to be written. The standard command form of a label followed by arguments separated by commas is assumed. *Command* may be a parameter assignment (e.g., ***CFWRITE**, A = 5).

Notes

Writes an ANSYS command (or similar string) to the file opened with ***CFOPEN**. The *Command* string is not executed (except that numeric and character parameter substitution and operations (with imbedded *, /, >, etc. characters) are performed before writing). When used with ***GET** results and parameter substitution, an ANSYS command can be created from results and then read back into the ANSYS program (or used elsewhere). For example, if the command ***CFWRITE**,BF,NNUM,TEMP,TVAL is used in a do-loop, where TVAL is a parameter value returned from the ***GET** operation and NNUM is a specified or returned parameter value, a series of **BF** commands, with numerical values substituted for the two parameters, will be written. To create a file without parameter substitution, use ***CREATE**.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

***CREATE**, *Fname*, *Ext*, --

Opens (creates) a macro file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Fname

File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Do not use a directory path if file is to be read with the macro *Name* option of the ***USE** command.

Ext

Filename extension (8 character maximum).

Ext should not be used if file is to be read with the macro *Name* option of the ***USE** command.

--

Unused field

Notes

See the ***USE** command for a discussion of macros. All commands following the ***CREATE** command, up to the ***END** command, are written to the specified file without being executed. An existing file of the same name, if any, will be overwritten. Parameter values are not substituted for parameter names in the commands when the

commands are written to the file. Use ***CFWRITE** to create a file if this is desired. The resulting macro may be executed with a ***USE** command (which also allows parameters to be passed into the macro) or a **/INPUT** command (which does not allow parameters to be passed in). Several macros may be stacked into a library file [***ULIB**]. You cannot use ***CREATE** within a DO loop.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Create Macro

---

*CYCLE

Bypasses commands within a do-loop.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Notes

Bypasses all commands between this command and the ***ENDDO** command within a do-loop. The next loop (if applicable) is initiated. The cycle option may also be conditionally executed [Use the ***IF**]. The ***CYCLE** command must appear on the same file as the ***DO** command.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*DEL, *Val1*, *Val2*

Deletes a parameter or parameters (GUI).

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Val1

Val1 can be:

ALL --
: Indicates that you want to delete all user-defined parameters, or both all user-defined and all system parameters, as indicated by the *Val2* argument.

(blank) --
: Indicates that you want to delete the parameter(s) indicated by *Val2*.

Val2

Val2 can be:

LOC --
When *Val1* is (blank), use *Val2* to specify the location of the parameter within the **Array Parameters** dialog box. The location number is based on an alphabetically ordered list of all parameters in the database. Not valid when *Val1* is ALL.

_PRM --
When *Val1* is ALL, specifying _PRM for *Val2* deletes all parameters, including those named with an initial underbar (_) (except _STATUS and _RETURN). When *Val1* is (blank), specifying _PRM for *Val2* deletes only those parameters named with an initial underbar (_) (except _STATUS and _RETURN).

PRM_ --
When *Val1* is (blank), specifying PRM_ for *Val2* deletes only those parameters named with a trailing underbar (_). Not valid when *Val1* is ALL.

(blank) --
When *Val1* is ALL, specifying (blank) for *Val2* causes all user-defined parameters to be deleted.

Notes

This is a command generally created by the Graphical User Interface (GUI). It will appear in the log file (**Jobname.LOG**) if an array parameter is deleted from within the **Array Parameters** dialog box.

To delete all user-defined parameters, issue the command ***DEL**,ALL. To delete only those user-defined parameters named with a trailing underbar, issue the command ***DEL**,,PRM_. To delete all user-defined and all system parameters (except for _STATUS and _RETURN), issue the command ***DEL**,ALL,_PRM. To delete a parameter by specifying its location within the **Array Parameters** dialog box, issue the command ***DEL**,,LOC.

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Preprocessor>LS-DYNA Options>Inertia Options>Define Inertia
Main Menu>Solution>Define Loads>Delete>Structural>Section

/DFLAB, *DOF*, *DispLab*, *ForceLab*

Changes DOF labels for user custom elements.

APDL: Macro Files

MP ME ST DY <> PR EM EH FL PP ED

DOF
Number indicating which DOF is to have its labels changed. For example:

1 = UX,FX
2 = UY,FY
3 = UX,FZ
4 = ROTX,MX
etc.

DispLab
New label (4 character maximum) for the displacement label. The old label is no longer valid.

ForceLab
New label (4 character maximum) for the force label for this degree-of-freedom. The old label is no longer valid.

Notes

The **/DFLAB** command is rarely used. It is designed for users who are writing custom elements for ANSYS and want to use degrees-of-freedom that are not part of the standard ANSYS set.

Menu Paths

This command cannot be accessed from a menu.

---

*DIM, *Par*, *Type*, *IMAX*, *JMAX*, *KMAX*, *Var1*, *Var2*, *Var3*, *CSYSID*

Defines an array parameter and its dimensions.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par
Name of parameter to be dimensioned. See ***SET** for name restrictions.

Type
Array type:

ARRAY --
Arrays are similar to standard FORTRAN arrays (indices are integers) (default). Index numbers for the rows, columns, and planes are sequential values beginning with one. Used for 1-, 2-, or 3-D arrays.

ARR4 --
Same as ARRAY, but used to specify 4-D arrays.

ARR5 --
Same as ARRAY, but used to specify 5-D arrays.

CHAR --
Array entries are character strings (up to 8 characters each). Index numbers for rows, columns, and planes are sequential values beginning with one.

TABLE --
Array indices are real (non-integer) numbers which must be defined when filling the table. Index numbers for the rows and columns are stored in the zero column and row "array elements" and are initially assigned a near-zero value. Index numbers must be in ascending order and are used only for retrieving an array element. When retrieving an array element with a real index that does not match a specified index, linear interpolation is done among the nearest indices and the corresponding array element values [***SET**]. Used for 1-, 2-, or 3-D tables.

TAB4 --
Same as TABLE, but used to specify 4-D tables.

TAB5 --
Same as TABLE, but used to specify 5-D tables.

STRING --
Array entries are character strings (up to IMAX each). Index numbers for columns and planes are sequential values beginning with 1. Row index is character position in string.

IMAX
Extent of first dimension (row) (Limit 128 for strings). Defaults to 1.

JMAX
Extent of second dimension (column). Defaults to 1.

KMAX
Extent of third dimension (plane). Defaults to 1.

Var1
Variable name corresponding to the first dimension (row) for *Type* = TABLE. Defaults to Row.

Var2
Variable name corresponding to the second dimension (column) for *Type* = TABLE. Defaults to Column.

Var3
Variable name corresponding to the third dimension (plane) for *Type* = TABLE. Defaults to Plane.

CSYSID
An integer corresponding to the coordinate system ID Number.

Notes

Up to three dimensions (row, column, and plane) may be defined using ARRAY and TABLE. Use ARR4, ARR5, TAB4, and TAB5 to define up to five dimensions (row, column, plane, book, and shelf). An index number is associated with each row, column, and plane. For array and table type parameters, element values are initialized to zero. For character and string parameters, element values are initialized to (blank). A defined parameter must be deleted [***SET**] before its dimensions can be changed. Scalar (single valued) parameters should not be dimensioned. ***DIM**,A,,3 defines a vector array with elements A(1), A(2), and A(3). ***DIM**,B,,2,3 defines a 2x3 array with elements B(1,1), B(2,1), B(1,2), B(2,2), B(1,3), and B(2,3). Use ***STATUS**,*Par* to display elements of array *Par*. You can write formatted data files (tabular formatting) from data held in arrays through the ***VWRITE** command.

If you use table parameters to define boundary conditions, then *Var1*, *Var2*, and/or *Var3* can either specify a primary variable (listed in *DIM - Primary Variables) or can be an independent parameter. If specifying an independent parameter, then you must define an additional table for the independent parameter. The additional table must have the same name as the independent parameter and may be a function of one or more primary variables or another independent parameter. All independent parameters must relate to a primary variable.

Tabular load arrays can be defined in both global Cartesian (defalut) or local (see below) coordinate systems by specifying *CSYSID*, as defined in **LOCAL**. For batch operations, you must specify your coordinate system first.

The following constraints apply when you specify a local coordinate system for your tabular loads:

Only Cartesian, cylindrical and spherical coordinate systems are supported
Angle values for THETA in cylindrical coordinate system must be input in degrees and must be positive values between 0 and 360 degrees ($0 \leq \Theta = 360$)
Angle values for THETA in spherical coordinate system must be input in degrees and must be positive values between 0 and 360 degrees ($0 \leq \Theta = 360$)
Angle values for Φ in spherical coordinate system must be input in degrees and must be positive values between -90 and +90 ($-90 \leq \Phi \leq 90$)

If you are specifying a 4- or 5-D array or table, four additional fields (*LMAX*, *MMAX*, *Var4*, and *Var5*) are available. Thus, for a 4-D table, the command syntax would be:

```
*DIM,Par,Type,IMAX,JMAX,KMAX,LMAX,Var1,Var2,Var3,Var4,CSYSID
```

For a 5-D table, the command syntax would be:

```
*DIM,Par,Type,IMAX,JMAX,KMAX,LMAX,MMAX,Var1,Var2,Var3,Var4,Var5,CSYSID
```

You cannot create or edit 4- or 5-D arrays or tables using the GUI.

See Section 3.11: Array Parameters for a detailed discussion on and examples for using array parameters.

*DIM - Primary Variables

Primary Variable	Label for *Var1, Var2, Var3*
Time	TIME
X-coordinate location	X
Y-coordinate location	Y
Z-coordinate location	Z
Temperature	TEMP
Velocity	VELOCITY
Pressure	PRESSURE
Cyclic sector number	SECTOR

Note — The X, Y, and Z coordinate locations listed above are valid in gobal Cartesian, or local (Cartesian, cylindrical and spherical) coordinate systems. The VELOCITY label is applicable only to the calculated fluid velocity in element FLUID116.

If you use table parameters to define boundary conditions, the table names (*Par*) must not exceed 32 characters.

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Settings>Replace vs Add>Smooth Data
Main Menu>Preprocessor>Loads>Load Step Opts>Time/Frequenc>Time - Time Step
Main Menu>Preprocessor>Loads>Load Step Opts>Time/Frequenc>Time and Substps
Main Menu>Preprocessor>LS-DYNA Options>Inertia Options>Define Inertia
Main Menu>Preprocessor>LS-DYNA Options>Loading Options>Smooth Data
Main Menu>Solution>Define Loads>Settings>Replace vs Add>Smooth Data
Main Menu>Solution>Load Step Opts>Time/Frequenc>Time - Time Step
Main Menu>Solution>Load Step Opts>Time/Frequenc>Time and Substps
Main Menu>Solution>Loading Options>Smooth Data
Main Menu>TimeHist Postpro>Smooth Data
Utility Menu>Parameters>Array Parameters>Define/Edit

/DIRECTORY, *StrArray*, *FileName*, *Ext*, *Dir*

Put the file names in the current directory into a string parameter array.

APDL: Array Parameters

MP ME ST DY <> PR EM EH FL PP ED

StrArray
Name of the "string array" parameter which will hold the returned values. String array parameters are similar to character arrays, but each array element can be as long as 128 characters. If the string parameter does not exist, it will be created. The array will be created as: *DIM,StrArray,STRING,64,2,numFileName

FileName
File name (64 characters maximum). Only files matching this name will be returned. The *FileName* ALL may match any file name.

Ext
File name extension (8 characters maximum). Only files with an extension matching this name will be returned. A blank or ALL will match any extension.

Directory
The directory in which the files reside. The default is the current working directory.

Notes

The **/DIRECTORY** command gets the file names in the current directory and puts them into a string parameter array. Each file will be included in the array as a name-extension pair.

Menu Paths

This command cannot be accessed from a menu.

***DO**, *Par*, *IVAL*, *FVAL*, *INC*

Defines the beginning of a do-loop.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Par
The name of the scalar parameter to be used as the loop index. See ***SET** for name restrictions. Any existing parameter of the same name will be redefined. There is no character parameter substitution for the *Par* field.

IVAL, *FVAL*, *INC*
Initially assign *IVAL* to *Par*. Increment *IVAL* by *INC* for each successive loop. If *IVAL* exceeds *FVAL* and *INC* is positive, the loop is not executed. *INC* defaults to 1. Negative increments and non-integer numbers are allowed.

Notes

The block of commands following the ***DO** command (up to the ***ENDDO** command) is executed repeatedly until some loop control is satisfied. Printout is automatically suppressed on all loops after the first (include a **/GOPR** command to restore the printout). The command line loop control (*Par*,*IVAL*,*FVAL*,*INC*) must be input;

however, a Use the ***IF** within the block can also be used to control looping [***EXIT** , ***CYCLE**]. One level of internal file switching is used for each nested ***DO**. Twenty levels of nested do-loops are allowed.

Note — Do-loops that include **/INPUT**, ***USE**, or an "Unknown Command" macro, have less nesting available because each of these operations also uses a level of file switching. The ***DO**, ***ENDDO**, and any ***CYCLE** and ***EXIT** commands for a do-loop must all be read from the same file (or keyboard). You cannot use the **MULTIPRO** or ***CREATE** commands within a ***DO**-loop. Picking operations should also not be used within a ***DO**-loop.

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Prob Design>Prob Method>Response Surface
Main Menu>Solution>Define Loads>Delete>Structural>Section

***DOWHILE**, *Par*

Loops repeatedly through the next *ENDDO command.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Par

The name of the scalar parameter to be used as the loop index. There is no character parameter substitution for the *Par* field.

Notes

DOWHILE** loops repeatedly through the next ***ENDDO** command as long as *Par* is greater than zero. The block of commands following the ***DOWHILE** command (up to the ***ENDDO** command) is executed repeatedly until some loop control is satisfied. Printout is automatically suppressed on all loops after the first (include a **/GOPR** command to restore the printout). The command line loop control (*Par*) must be input; however, ***IF** within the block can also be used to control looping [EXIT**, ***CYCLE**]. One level of internal file switching is used for each nested ***DOWHILE**. Twenty levels of nested do-loops are allowed.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*ELSE

Separates the final if-then-else block.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Notes

Optional final block separator within an if-then-else construct. See the ***IF** for details. If a batch input stream hits an end-of-file during a false ***IF** condition, the ANSYS run will not terminate normally. You will need to terminate it externally (use either the UNIX "kill" function or the Windows task manager). The ***ELSE** command must appear on the same file as the ***IF** command, and all five characters must be input.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*ELSEIF, *VAL1*, *Oper*, *VAL2*

Separates an intermediate if-then-else block.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

VAL1
First numerical value (or parameter which evaluates to numerical value) in the conditional comparison operation.

Oper1
Operation label. A tolerance of 1.0E-10 is used for comparisons between real numbers:

EQ --
Equal (for *VAL1* = *VAL2*).

NE --
Not equal (for *VAL1* $\neq$ *VAL2*).

LT --
Less than (for *VAL1*<*VAL2*).

GT --
Greater than (for *VAL1*>*VAL2*).

LE --
Less than or equal (for *VAL1* $\leq$ *VAL2*).

GE --
Greater than or equal (for *VAL1* $\geq$ *VAL2*).

ABLT --
Absolute values of *VAL1* and *VAL2* before < operation.

ABGT --
Absolute values of *VAL1* and *VAL2* before > operation.

VAL2
Second numerical value (or parameter which evaluates to numerical value) in the conditional comparison operation.

Conj
(Optional) Connection between two logical clauses.

AND -
True if both clauses are true.

OR -
True if either clause is true.

XOR -
True if either (but not both) clause is true.

VAL3
(Optional) Third numerical value (or parameter which evaluates to numerical value).

Oper2
(Optional) Operation label. A tolerance of 1.0E-10 is used for comparisons between real numbers.

VAL4
(Optional) Fourth Numerical value (or parameter value which evaluates to a numerical value).

Notes

Optional intermediate block separator within an if-then-else construct. *VAL1* and *VAL2* can also be character strings (enclosed in quotes) or parameters for *Oper* = EQ and NE only. All seven characters must be input. Similar to Use the ***IF** except that the *Base* field is not used. The ***ELSEIF** command must appear on the same file as the Use the ***IF** command.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*END

Closes a macro file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Notes

Closes a file opened with ***CREATE**. The ***END** command is an 8-character command (to differentiate it from ***ENDIF**). If you add commented text on that same line but do not allow enough spaces between ***END** and the "!" that indicates the comment text, the ***END** will attempt to interpret the "!" as the 8th character and will fail.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*ENDDO

Ends a do-loop and starts the looping action.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Notes

One ***ENDDO** is required for each nested do-loop. The ***ENDDO** command must appear on the same file as the ***DO** command, and all six characters must be input.

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Prob Design>Prob Method>Response Surface
Main Menu>Solution>Define Loads>Delete>Structural>Section

---

*ENDIF

Ends an if-then-else.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Notes

Required terminator for the if-then-else construct. See the ***IF** for details. If a batch input stream hits an end-of-file during a false ***IF** condition, the ANSYS run will not terminate normally. You will need to terminate it externally (use either the UNIX "kill" function or the Windows task manager). The ***ENDIF** command must appear on the same file as the ***IF** command, and all six characters must be input.

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Solution>Define Loads>Delete>Structural>Section

---

*EXIT

Exits a do-loop.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Notes

The command following the ***ENDDO** is executed next. The exit option may also be conditional [Use the ***IF**]. The ***EXIT** command must appear on the same file as the ***DO** command.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*GET, *Par*, *Entity*, *ENTNUM*, *Item1*, *IT1NUM*, *Item2*, *IT2NUM*

Retrieves a value and stores it as a scalar parameter or part of an array parameter.

APDL: Parameters

MP ME ST DY <> <> EM <> FL PP ED

Par

The name of the resulting parameter. See ***SET** for name restrictions.

Entity

Entity keyword. Valid keywords are NODE, ELEM, KP, LINE, AREA, VOLU, PDS, etc., as shown for *Entity* = in the tables below.

ENTNUM

The number or label for the entity (as shown for *ENTNUM* = in the tables below). In some cases, a zero (or blank) *ENTNUM* represents all entities of the set.

Item1

The name of a particular item for the given entity. Valid items are as shown in the *Item1* columns of the tables below.

IT1NUM

The number (or label) for the specified *Item1* (if any). Valid *IT1NUM* values are as shown in the *IT1NUM* columns of the tables below. Some *Item1* labels do not require an *IT1NUM* value.

Item2, *IT2NUM*

A second set of item labels and numbers to further qualify the item for which data are to be retrieved. Most items do not require this level of information.

Notes

***GET** retrieves a value for a specified item and stores the value as a scalar parameter, or as a value in a user-named array parameter. An item is identified by various keyword, label, and number combinations. Usage is similar to the ***SET** command except that the parameter values are retrieved from previously input or calculated results. For example, , A, ELEM, 5, CENT, X returns the centroid x-location of element 5 and stores the result as

parameter A. ***GET** command operations, along with the associated Get functions return values in the active coordinate system unless stated otherwise.

Both ***GET** and ***VGET** retrieve information from the active data stored in memory. The database is often the source, and sometimes the information is retrieved from common memory blocks that ANSYS uses to manipulate information. Although POST1 and POST26 operations use a ***.rst** file, ***GET** data is accessed from the database or from the common blocks. Get operations do not access the ***.rst** file directly. For repeated gets of sequential items, such as from a series of elements, see the ***VGET** command.

Most items are stored in the database after they are calculated and are available anytime thereafter. Items are grouped according to where they are usually first defined or calculated. Most of the GENERAL items listed below are available from all modules. Each of the sections for accessing *GET parameters are shown in the following order:

- *GET General Entity Items
- *GET Preprocessing Entity Items
- *GET Solution Entity Items
- *GET Postprocessing Entity Items
- *GET Optimization and Probabilistic Design Entity Items

The ***GET** command is valid in any processor.

General Items

*GET General Entity Items

- *GET General Items, Entity = ACTIVE
- *GET General Items, Entity = CMD
- *GET General Items, Entity = COMP
- *GET General Items, Entity = GRAPH
- *GET General Items, Entity = PARM

*GET General Items, Entity = ACTIVE

***Entity* = ACTIVE, *ENTNUM* = 0 (or blank)**		
***GET, Par, ACTIVE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
INT		Current interactive key: 0=off, 2=on.
IMME		Current immediate key: 0=off, 1=on.
MENU		Current menu key: 0=off, 1=on.
PRKEY		Printout suppression status: 0=**/NOPR**, 1=**/GOPR** or **/GO**
UNITS		Units specified by **/UNITS** command: 0 = USER, 1 = SI, 2 = CGS, 3 = BFT, 4 = BIN, 6 = MPA.
ROUT		Current routine: 0 = Begin level, 17 = PREP7, 21 = SOLUTION, 31 = POST1, 36 = POST26, 41 = OPT, 52 = AUX2, 62 = AUX12, 65 = AUX15, 71 = RUNSTAT.

***Entity* = ACTIVE, *ENTNUM* = 0 (or blank)**		
***GET, Par, ACTIVE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TIME	WALL,CPU	Current wall clock or CPU time. Current wall clock will continue to accumulate during an ANSYS run and is NOT reset to zero at midnight.
DBASE	LDATE	Date of first modification of any database quantity required for POST1 operation. The parameter returned is *Par* = YEAR*10000 + MONTH*100 + DAY.
DBASE	LTIME	Time of last modification of any database quantity required for POST1 operation. The parameter returned is *Par* = HOURS*10000 + MINUTES*100 + SECONDS.
REV		ANSYS minor revision number (5.6, 5.7, 6.0 etc.). Letter notation (e.g., 5.0A) is not included.
TITLE	0,1,2,3,4	**Item2:** START **IT2NUM:***N* Current title string of the main title (*IT1NUM*=0 or blank) or subtitle 1, 2, 3, or 4 (*IT1NUM*=1,2,3, or 4). A character parameter of up to 8 characters, starting at position *N*, is returned.
JOBNAM		**Item2:** START **IT2NUM:***N* Current Jobname. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all 32 characters.
PLATFORM		The current platform.
NPROC		The maximum number of processors available.

*GET General Items, Entity = CMD

***Entity* = CMD, *ENTNUM* = 0 (or blank)** **The following items are valid for all commands except star (*) commands and non-graphics slash (/) commands.**		
***GET, *Par*, CMD, 0, Item1, IT1NUM, Item2, IT2NUM.**		
Item1	**IT1NUM**	**Description**
STAT		Status of previous command: 0=found, 1=not found (unknown).
NARGS		Field number of last nonblank field on the previous command.
FIELD	2,3...*N*	Numerical value of the *N*th field on the previous command. Field 1 is the command name (not available)

*GET General Items, Entity = COMP

***Entity* = COMP, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, COMP, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NCOMP		Total number of components and assemblies currently defined.
***Entity* = COMP, *ENTNUM* = *n* (*n*th component)**		
***GET, *Par*, COMP, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NAME		Name of the *N*th item (component or assembly) in the list of components and assemblies. A character parameter is returned.

***Entity* = COMP, *ENTNUM* = *Cname* (component or assembly name)**		
***GET, *Par*, COMP, *Cname*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TYPE		Type of component *Cname*: 1=Nodes, 2=Elements, 6=Keypoints, 7=Lines, 8=Areas, 9=Volumes, 11-15=Subcomponents (11=subcomponent at level 1, 12=subcomponent at level 2, etc.).
NSCOMP		Number of subcomponents (for assemblies).
SNAME	*N*	Name of *N*th subcomponent of assembly *Cname*. A character parameter is returned.

*GET General Items, Entity = GRAPH

***Entity* =GRAPH, *ENTNUM* = N (window number)**		
***GET, *Par*, GRAPH, N, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ACTIVE		**/WINDOW** status: 0=off, 1=on.
ANGLE		**/ANGLE***THETA* angle.
CONTOUR	*Name*	**/CONTOUR** value for *Name*, where *Name* = VMIN, VINC, or NCONT.
DIST		**/DIST***DVAL* value.
DSCALE	DMULT	**/DSCALE** *DMULT* value.
EDGE		**/EDGE***KEY* value.
FOCUS	X, Y, Z	**/FOCUS***XF*, *YF*, or *ZF* value.
GLINE		**/GLINE***STYLE* value.
MODE		**/USER** or **/AUTO** setting: 0=user, 1=auto.
NORMAL		**/NORMAL***KEY* value.
RANGE	XMIN, XMAX, YMIN, YMAX	**/WINDOW***XMIN*, *XMAX*, *YMIN* , or *YMAX* screen coordinates.
RATIO	X, Y	**/RATIO***RATOX* or *RATOY*value.
SSCALE	SMULT	**/SSCALE***SMULT* value.
TYPE		**/TYPE***Type* value.
VCONE	ANGLE	**/VCONE***PHI* angle.
VIEW	X, Y, Z	**/VIEW***XV*, *YV*, or *ZV* value.
VSCALE	VRATIO	**/VSCALE***VRATIO* value.
***Entity* =GRAPH, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, GRAPH, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
DISPLAY		**/SHOW***VECT* setting: 0=raster, 1=vector.
ERASE		**/ERASE** or **/NOERASE** setting: 0=no erase, 1=erase.
NDIST		Largest nodal range for current model (DX, DY, or DZ of the model).
NUMBER		**/NUMBER***NKEY* value.
PLOPTS	*Name*	**/PLOPTS** setting of *Name*, where *Name*=LEG1, LEG2, LEG3, INFO, FRAM, TITL, MINM, or VERS.

Entity =GRAPH, *ENTNUM* = 0 (or blank)		
***GET, *Par*, GRAPH, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SEG		Segment capability of graphics driver: 0=no segments available, 1=erasable segments available, 2=non-erasable segments available.
SHRINK		**/SHRINK***RATIO* value.

*GET General Items, Entity = PARM

Entity, = PARM*ENTNUM* = 0 (or blank)		
***GET, *Par*, PARM, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MAX		Total number of parameters currently defined.
BASIC		Number of scalar parameters (excluding parameters beginning with an underscore _, array parameters, and character parameters).
LOC	*Num*	Name of the parameter at the *Num* location in the parameter table. A character parameter is returned.
Entity = PARM, *ENTNUM* = Name (parameter name)		
***GET, *Par*, PARM, *Name*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TYPE		Parameter type: 0=scalar, 1=array, 2=table, 3=character scalar, 4=character array, -1=undefined
DIM	X,Y,Z	Row (X), Column (Y), or Plane (Z) dimension of array parameter.

Preprocessing Items

*GET Preprocessing Entity Items

- *GET Preprocessing Items, Entity = ACTIVE
- *GET Preprocessing items, Entity = AREA
- *GET Preprocessing Items, Entity = CDSY
- *GET Preprocessing Items, Entity = CE
- *GET Preprocessing Items, Entity = CP
- *GET Preprocessing Items, Entity = EDCC
- *GET Preprocessing Items, Entity = ELEM
- *GET Preprocessing Items, Entity = ETYP
- *GET Preprocessing Items, Entity = FLDATA
- *GET Preprocessing Items, Entity = KP
- *GET Preprocessing Items, Entity = LINE
- *GET Preprocessing Items, Entity = MPLAB
- *GET Preprocessing Items, Entity = MSCAP
- *GET Preprocessing Items, Entity = MSDATA

- *GET Preprocessing Items, Entity = MSMETH
- *GET Preprocessing Items, Entity = MSNOMF
- *GET Preprocessing Items, Entity = MSPROP
- *GET Preprocessing Items, Entity = MSRELAX
- *GET Preprocessing Items, Entity = MSSOLU
- *GET Preprocessing Items, Entity = MSSPEC
- *GET Preprocessing Items, Entity = MSVARY
- *GET Preprocessing Items, Entity = NODE
- *GET Preprocessing Items, Entity = PART
- *GET Preprocessing Items, Entity = RCON
- *GET Preprocessing Items, Entity = SCTN
- *GET Preprocessing Items, Entity = SECP
- *GET Preprocessing Items, Entity = SHEL
- *GET Preprocessing Items, Entity = TBFT
- *GET Preprocessing Items, Entity = TBLAB
- *GET Preprocessing Items, Entity = VOLU
- *GET Preprocessing Items, Entity = WELD

*GET Preprocessing Items, Entity = ACTIVE

***Entity* = ACTIVE, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, ACTIVE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SEG		Segment capability of graphics driver: 0=no segments available, 1=erasable segments available, 2=non-erasable segments available.
CSYS		Active coordinate system.
DSYS		Active display coordinate system.
MAT		Active material.
TYPE		Active element type.
REAL		Active real constant set.
ESYS		Active element coordinate system.
SECT		Active section.
CP		Maximum coupled node set number in the model (includes merged and deleted sets until compressed out).
CE		Maximum constraint equation set number in the model (includes merged and deleted sets until compressed out).
WFRONT	MAX, RMS	Current maximum or RMS wavefront. Zero if no reordering done.

*GET Preprocessing items, Entity = AREA

***Entity* = AREA, *ENTNUM* = *N* (area number)**		
***GET, *Par*, AREA, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ATTR	*Name*	Number assigned to the attribute, *Name*, where *Name*=MAT, TYPE, REAL, ESYS, SECN, NNOD, NELM, or ESIZ. (NNOD=number of nodes, NELM=number of elements, ESIZ=element size.)
ASEL		Select status of area *N*: -1=unselected, 0=undefined, 1=selected. Alternative get function: ASEL(*N*).
NXTH		Next higher area number above *N* in selected set (or zero if none found).
NXTL		Next lower area number below *N* in selected set (or zero if none found).
AREA		Area of area *N*. (**ASUM** or **GSUM** must have been performed sometime previously with at least this area *N* selected).
LOOP	1,2,...,*I*	*Item2*: LINE, *IT2NUM*: 1,2,...,*p* Line number of position *p* of loop *I*
***Entity* = AREA, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, AREA, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX, MIN	Highest or lowest area number in the selected set.
NUM	MAXD, MIND	Highest or lowest area number defined.
COUNT		Number of areas in the selected set.
AREA		Combined areas (from last **ASUM** or **GSUM**).
VOLU		Combined volume of areas (from last **ASUM** or **GSUM**. For 3-D area elements, thickness is determined from area attributes [**AATT**]. For 2-D elements, area attributes are ignored and unit thickness is assumed.
CENT	X, Y, Z	Centroid X, Y, or Z location of areas (from last **ASUM** or **GSUM**).
IOR	X, Y, Z, XY, YZ, ZX	Moments of inertia about origin (from last **ASUM** or **GSUM**).
IMC	X, Y, Z, XY, YZ, ZX	Moments of inertia about mass centroid (from last **ASUM** or **GSUM**).
IPR	X, Y, Z	Principal moments of inertia (from last **ASUM** or **GSUM**).
IXV	X, Y, Z	Principal orientation X vector components (from last **ASUM** or **GSUM**).
IYV	X, Y, Z	Principal orientation Y vector components (from last **ASUM** or **GSUM**).
IZV	X, Y, Z	Principal orientation Z vector components (from last **ASUM** or **GSUM**).

*GET Preprocessing Items, Entity = CDSY

***Entity* = CDSY, *ENTNUM* = *N* (coordinate system number)**		
***GET, *Par*, CDSY, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
LOC	X, Y, Z	X, Y, or Z origin location in global Cartesian system.
ANG	XY, YZ, ZX	THXY, THYZ, or THZX rotation angle (in degrees) relative to the global Cartesian coordinate system.
ATTR	*Name*	Number assigned to *Name*, where *Name*=KCS, KTHET, KPHI, PAR1, or PAR2. The value -1.0 is returned for KCS if the coordinate system is undefined.

***Entity* = CDSY, *ENTNUM* = *N* (coordinate system number)**		
***GET, *Par*, CDSY, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX	The maximum coordinate system number

*GET Preprocessing Items, Entity = CE

***Entity* = CE, *ENTNUM* = *N* (constraint equation set)**		
***GET, *Par*, CE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
If N = 0, then		
MAX		Maximum constraint equation number
NUM		Number of constraint equations
If N > 0, then		
NTERM		Number of terms in this constraint equation
CONST		Constant term for this constraint equation
TERM	number	Item2 = NODE: Gives the node for this position in the constraint equation. Item2 = DOF: Gives the DOF number for this position in the constraint equation. (1–UX, 2–UY, 3–UZ, 4–ROTX, etc.) Item2 = COEF: Gives the coefficient for this position in the constraint equation.

*GET Preprocessing Items, Entity = CP

***Entity* = CP, *ENTNUM* = *N* (coupled node set)**		
***GET, *Par*, CP, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
If N = 0, then		
MAX		Maximum coupled set number
NUM		Number of coupled sets
If N > 0, then		
DOF		The degree of freedom for this set (1–UX, 2–UY, 3–UZ, 4–ROTX, etc.)
NTERM		Number of nodes in this set.
TERM	number	Item2 = NODE: Gives the node for this position number in the coupled set.

*GET Preprocessing Items, Entity = EDCC

***Entity* = EDCC, *ENTNUM* = *N* (contact entity number, obtained by issuing the EDCLIST command)**		
***GET, *Par*, EDCC, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
COMP	1, 2	Component name for the contact (1) or target (2) surface of contact entity *N*. A character parameter is returned.
PART	1, 2	PART number for contact (1) or target (2) surface of contact entity *N*.

Entity = EDCC, *ENTNUM* = 0 (or blank)		
***GET, *Par*, EDCC, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
COUNT		Total number of contact definitions.

*GET Preprocessing Items, Entity = ELEM

Entity = ELEM, *ENTNUM* = *N* (element number)		
***GET, *Par*, ELEM, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NODE	1, 2, ... 20	Node number at position 1,2,... or 20 of element*N*. Alternative get function: NELEM(*n*, *npos*), where *npos* is 1,2,...20.
CENT	X, Y, Z	Centroid X, Y, or Z location (based on shape function) in the active coordinate system. The original locations is used even if large deflections are active. Alternative get functions: CENTRX(*N*), CENTRY(*N*), and CENTRZ(*N*) always retrieve the element centroid in global Cartesian coordinates, and are determined from the selected nodes on the elements.
ADJ	1, 2, ... 6	Element number adjacent to face 1,2,...6. Alternative get function: ELADJ(*N*,*face*). Only elements (of the same dimensionality) adjacent to lateral faces are considered.
ATTR	*Name*	Number assigned to the attribute *Name*, where *Name* = MAT, TYPE, REAL, ESYS, PSTAT, LIVE, or SECN. Returns a zero if the element is unselected. If *Name* = PSTAT (valid for p-elements only), returns a 1 if the element is selected and included [**PINCLUDE**], and a -1 if the element is selected and excluded [**PEXCLUDE**]. If *Name* = LIVE, returns a 1 if the element is selected and active, and a -1 if it is selected and inactive. *Name* = SECN returns the section number of the selected beam element.
LENG		Length of line element (straight line between ends).
LPROJ	X, Y, Z	Projected line element length (in the active coordinate system). X is x-projection onto y-z plane, Y is y projection onto z-x plane, and Z is z-projection onto x-y plane.
AREA		Area of area element.
APROJ	X, Y, Z	Projected area of area element area (in the active coordinate system). X is x-projection onto y-z plane, Y is y projection onto z-x plane, and Z is z-projection onto x-y plane.
VOLU		Element volume. Based on unit thickness for 2-D plane elements (unless the thickness option is used) and on the full 360 degrees for 2-D axisymmetric elements. *Note* — If results data are in the database, the volume returned is the volume calculated during solution.
ESEL		Select status of element *N*: -1 = unselected, 0 = undefined, 1 = selected. Alternative get function: ESEL(*N*).
NXTH		Next higher element number above *N* in selected set (or zero if none found). Alternative get function: ELNEXT(*N*)
NXTL		Next lower element number below *N* in selected set (or zero if none found).
HGEN		Heat generation on selected element *N*.

***Entity* = ELEM, *ENTNUM* = *N* (element number)**		
***GET, *Par*, ELEM, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
HCOE	face	Heat coefficient for selected element *N* on specified face. Returns the value at the first node that forms the face.
TBULK	face	Bulk temperature for selected element *N* on specified face. Returns the value at the first node that forms the face.
PRES	face	Pressure on selected element, *N* on specified face. Returns the value at the first node that forms the face.
SHPAR	*Test*	Element shape test result for selected element *N*, where *Test* = ANGD (SHELL28 corner angle deviation), ASPE (aspect ratio), JACR (Jacobian ratio), MAXA (maximum corner angle), PARA (deviation from parallelism of opposite edges), or WARP (warping factor).
***Entity* = ELEM, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, ELEM, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX,MIN	Highest or lowest element number in the selected set.
NUM	MAXD, MIND	Highest or lowest element number defined.
COUNT		Number of elements in the selected set.

*GET Preprocessing Items, Entity = ETYP

***Entity* = ETYP, *ENTNUM* = *N* (element type number)**		
***GET, *Par*, ETYP, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ATTR	*Name*	Number assigned to the attribute *Name*, where *Name*=ENAM, KOP1, KOP2, ..., KOP9, KO10, KO11, etc.
***Entity* = ETYP, *ENTNUM* = 0 (or blank)**		
***GET,*Par*,ETYP,0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX	Maximum element type.

*GET Preprocessing Items, Entity = FLDATA

***Entity* = FLDATA, *ENTNUM* = *Name* (*Name* is a valid label on the *Name* field of the FLDATA command.)** **The value returned is the numerical value for numeric items, 0 or 1 for logical items (off/on or false/true), and a character parameter for items that require a character string. For example, *GET,X,FLDATA,TERM,PRES returns X=convergence monitor value for pressure [FLDATA3], *GET,X,FLDATA,SOLU,TURB returns X=1 if the turbulence model is ON [FLDATA1], and *GET,X,FLDATA,PROT,DENS returns X='CONSTANT' if density is specified as a constant property type [FLDATA7].**		
***GET, *Par*, FLDATA, *Name*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
Lab		Value of Lab, where Lab is a valid label from the *Label* field of the **FLDATA** command.

*GET Preprocessing Items, Entity = KP

***Entity* = KP, *ENTNUM* = *N* (keypoint number)**		
***GET, *Par*, KP, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
LOC	X, Y, Z	X, Y, or Z location in the active coordinate system. Alternative get functions: KX(*N*), KY(*N*), KZ(*N*). Inverse get function: KP(*x, y, z*) returns the number of the selected keypoint nearest the *x, y, z* location (in the active coordinate system, lowest number for coincident keypoints).
ATTR	*Name*	Number assigned to the attribute *Name*, where *Name* = MAT, TYPE, REAL, ESYS, NODE, or ELEM.
KSEL		Select status of keypoint *N*: -1 = unselected, 0 = undefined, 1 = selected. Alternative get function: KSEL(*N*).
NXTH		Next higher keypoint number above *N* in selected set (or zero if none found). Alternative get function: KPNEXT(*N*).
NXTL		Next lower keypoint number below *N* in selected set (or zero if none found).
DIV		Divisions (element size setting) from **KESIZE** command.
***Entity* = KP, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, KP, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX, MIN	Highest or lowest keypoint number in the selected set.
NUM	MAXD, MIND	Highest or lowest keypoint number defined
COUNT		Number of keypoints in the selected set.
CENT	X, Y, Z	Centroid X, Y, or Z location of keypoints (from last **KSUM** or **GSUM**).
IOR	X, Y, Z, XY, YZ, ZX	Moments of inertia about origin (from last **KSUM** or **GSUM**).
IMC	X, Y, Z, XY, YZ, ZX	Moments of inertia about mass centroid (from last **KSUM** or **GSUM**).
IPR	X, Y, Z	Principal moments of inertia (from last **KSUM** or **GSUM**).
IXV	X, Y, Z	Principal orientation X vector components (from last **KSUM** or **GSUM**).
IYV	X, Y, Z	Principal orientation Y vector components (from last **KSUM** or **GSUM**).
IZV	X, Y, Z	Principal orientation Z vector components (from last **KSUM** or **GSUM**).
MXLOC	X, Y, Z	Maximum X, Y, or Z keypoint coordinate in the selected set (in the active coordinate system).
MNLOC	X, Y, Z	Minimum X, Y, or Z keypoint coordinate in the selected set (in the active coordinate system).
NRELM	m	Keypoint number of meshed region nearest centroid of element *m*.

*GET Preprocessing Items, Entity = LINE

***Entity* = LINE, *ENTNUM* = *N* (line number)**		
***GET, *Par*, LINE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
KP	1,2	Keypoint number at position 1 or 2.

Entity = LINE, *ENTNUM* = *N* (line number)		
***GET, *Par*, LINE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ATTR	*Name*	Number assigned to the attribute, *Name*, where *Name*=MAT, TYPE, REAL, ESYS, NNOD, NELM, NDIV, NDNX, SPAC, SPNX, KYND, KYSP, LAY1, or LAY2. (NNOD=number of nodes, returns --1 for meshed line with no internal nodes, NELM=number of elements, NDIV=number of divisions in an existing mesh, NDNX=number of divisions assigned for next mesh, SPAC=spacing ratio in an existing mesh, SPNX=spacing ratio for next mesh, KYND=soft key for NDNX, KYSP=soft key for SPNX, LAY1=LAYER1 setting, LAY2=LAYER2 setting.)
LSEL		Select status of line *N*: -1=unselected, 0=undefined, 1=selected. Alternative get function: LSEL(*N*).
NXTH		Next higher line number above *N* in the selected set (or zero if none found). Alternative get function: LSNEXT(*N*)
NXTL		Next lower line number below *N* in selected set (or zero if none found).
LENG		Length. A get function LX(*n, lfrac*) also exists to return the X coordinate location of line *N* at the length fraction *lfrac* (0.0 to 1.0). Similar LY and LZ functions exist.
Entity = LINE, *ENTNUM* = 0 (or blank)		
***GET, *Par*, LINE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX, MIN	Highest or lowest line number in the selected set.
NUM	MIND, MAXD	Highest or lowest line number defined.
COUNT		Number of lines in the selected set.
LENG		Combined length of lines (from last **LSUM** or **GSUM**).
CENT	X, Y, Z	Centroid X, Y, or Z location of lines (from last **LSUM** or **GSUM**).
IOR	X, Y, Z, XY, YZ, ZX	Moments of inertia about origin (from last **LSUM** or **GSUM**).
IMC	X, Y, Z, XY, YZ, ZX	Moments of inertia about mass centroid (from last **LSUM** or **GSUM**).
IPR	X, Y, Z	Principal moments of inertia (from last **LSUM** or **GSUM**).
IXV	X, Y, Z	Principal orientation X vector components (from last **LSUM** or **GSUM**).
IYV	X, Y, Z	Principal orientation Y vector components (from last **LSUM** or **GSUM**).
IZV	X, Y, Z	Principal orientation Z vector components (from last **LSUM** or **GSUM**).

*GET Preprocessing Items, Entity = MPLAB

Entity = MPlab, *ENTNUM* =*N* (*MPlab* = material property label from MP command; *N* = material number.)		
***GET, *Par*, MPlab, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TEMP	val	Material property value at temperature of *val*. For temperature dependant materials, the program interpolates the property at temperature input for *val*.

*GET Preprocessing Items, Entity = MSCAP

Entity = MSCAP, *ENTNUM* = *n* (species number)		
***GET, *Par*, MSCAP, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
KEY		Status of mass fraction capping for species *n:* 0=off, 1=on.
UPP		Upper bound of mass fraction.
LOW		Lower bound of mass fraction.

*GET Preprocessing Items, Entity = MSDATA

Entity = MSDATA, *ENTNUM* = 0		
***GET, *Par*, MSDATA, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SPEC		The algebraic species number.
UGAS		Value of the universal gas constant.

*GET Preprocessing Items, Entity = MSMETH

Entity = MSMETH, *ENTNUM* = *n* (species number)		
***GET, *Par*, MSMETH, *n*, Item1, IT1NUM, Item2, IT2NUM**		
(Blank)	(Blank)	Solution method for species *n*: 0=no solution, 1=TDMA method, 2=conjugate residual method, 3=preconditioned conjugate residual method.

*GET Preprocessing Items, Entity = MSNOMF

Entity = MSNOMF, *ENTNUM* = *n* (species number)		
***GET, *Par*, MSNOMF, *n*, Item1, IT1NUM, Item2, IT2NUM**		
(Blank)	(Blank)	Initial mass fraction of species *N*.

*GET Preprocessing Items, Entity = MSPROP

Entity = MSPROP, *ENTNUM* = *n* (species number)		
***GET, *Par*, MSPROP, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
Lab	TYPE	The type of fluid property variation being used for *Lab*, where *Lab* is a valid property label as described on the **MSPROP** command (DENS, VISC, COND, etc.). A character parameter is returned: CONSTANT, GAS, LIQUID, or a property name from the floprp.ans file.
"	NOMI	Value of property *Lab:* nominal value for a CONSTANT fluid property, value at temperature given by *COF1* for other property types.
"	COF1, COF2, COF3	Coefficients in the equation of state for property *Lab*.

*GET Preprocessing Items, Entity = MSRELAX

***Entity* = MSRELAX, *ENTNUM* = *n* (species number)**		
***GET, *Par*, MSRELAX, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
CONC		Mass fraction concentration relaxation factor.
MDIF		Mass diffusion coefficient relaxation factor.
EMDI		Effective mass diffusion coefficient relaxation factor.
STAB		Transport equation inertial relaxation factor.

*GET Preprocessing Items, Entity = MSSOLU

***Entity* = MSSOLU, *ENTNUM* = *n* (species number)**		
***GET, *Par*, MSSOLU,*n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NSWE		Number of TDMA sweeps.
MAXI		Maximum number of iterations allowed for semi-direct methods.
NSRC		Number of search vectors used for semi-direct methods.
CONV		Convergence criterion for semi-direct methods.
DELT		Maximum normalized rate of change which will permit the semi-direct solution to continue.

*GET Preprocessing Items, Entity = MSSPEC

***Entity* = MSSPEC, *ENTNUM* = *n* (species number)**		
***GET, *Par*, MSSPEC, *n*, Item1, IT1NUM, Item2, IT2NUM**		
NAME		Name of species *n*. A character parameter is returned.
MOLW		Molecular weight of species *n*.
SCHM		Turbulent Schmidt number of species *n*.

*GET Preprocessing Items, Entity = MSVARY

***Entity* = MSVARY, *ENTNUM* = *n* (species number)**		
***GET, *Par*, MSVARY, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
Lab		Variability status of property *Lab* (where *Lab* =DENS, VISC, COND, or MDIF): 0=off, 1=on.

*GET Preprocessing Items, Entity = NODE

***Entity* = NODE, *ENTNUM* = *N* (node number)**		
***GET, *Par*, NODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
LOC	X, Y, Z	X, Y, Z location in the active coordinate system. Alternative get functions: NX(*N*), NY(*N*), NZ(*N*). Inverse get function. NODE(*x, y, z*) returns the number of the selected node nearest the *x, y, z* location (in the active coordinate system, lowest number for coincident nodes).
ANG	XY, YZ, ZX	THXY, THYZ, THZX rotation angle.
NSEL		Select status of node *N*: -1=unselected, 0=undefined, 1=selected. Alternative get function: NSEL(*N*).
NXTH		Next higher node number above *N* in selected set (or zero if none found). Alternative get function: NDNEXT(*N*).
NXTL		Next lower node number below *N* in selected set (or zero if none found).
F	FX, MX, ...	Applied force at selected node *N* in direction *IT1NUM* (returns 0.0 if no force is defined, if node is unselected, or if the DOF is inactive). If *ITEM2* is IMAG, return the imaginary part.
D	UX, ROTX, ...	Applied constraint force at selected node *N* in direction *IT1NUM* (returns a large number, such as 2e100, if no constraint is specified, if the node is unselected, or if the DOF is inactive). If *ITEM2* is IMAG, return the imaginary part.
HGEN		Heat generation on selected node *N* (returns 0.0 if node is unselected, or if the DOF is inactive).
NTEMP		Temperature on selected node N (returns 0.0 if node is unselected)
CPS	Lab	Couple set number with direction Lab = any active DOF, which contains the node *N*.
***Entity* = NODE, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, NODE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX, MIN	Highest or lowest node number in the selected set.
NUM	MAXD, MIND	Highest or lowest node number defined.
COUNT		Number of nodes in the selected set.
MXLOC	X, Y, Z	Maximum X, Y, or Z node coordinate in the selected set (in the active coordinate system).
MNLOC	X, Y, Z	Minimum X, Y, or Z node coordinate in the selected set (in the active coordinate system).
Note: If internal nodes are created during solution by ANSYS, the internal nodes may also be included by the command. You can select/unselect them by node numbers, but they can not be listed or plotted.		

*GET Preprocessing Items, Entity = PART

***Entity* = PART, *ENTNUM* = *N* (PART number)**		
***GET, *Par*, PART, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TYPE		Element type number assigned to PART *N*.
MAT		Material number assigned to PART *N*.

***Entity* = PART, *ENTNUM* = *N* (PART number)**		
***GET, *Par*, PART, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
REAL		Real constant number assigned to PART *N*.
***Entity* = PART, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, PART, 0, Item1, IT1NUM, Item2, IT2NUM**		
NUMP		Total number of parts in the model.

*GET Preprocessing Items, Entity = RCON

***Entity* = RCON, *ENTNUM* = *N* (real constant set number)**		
***GET, *Par*, RCON, *N*, Item1, IT1NUM, Item2, IT2NUM**		
CONST	1, 2, ..., *m*	Value of real constant number *m* in set *N*.
NUM	MAX	The maximum real constant number

*GET Preprocessing Items, Entity = SCTN

***Entity* = SCTN, *ENTNUM* = *N* (pretension section ID number)**		
***GET, *Par*, SCTN, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
1		Section ID number.
2		Section type (always 5 for pretension section).
3		Pretension node number.
4	Coordinate system number.	Section normal NX.
5	Coordinate system number.	Section normal NY.
6	Coordinate system number.	Section normal NZ.
7 or 8		Eight character section name.
9		Initial action key. Returns 0 or 1 for lock, 2 for "free-to-slide," or 3 for tiny.
10		Force displacement key. Returns 0 or 1 for force, or 2 for displacement.
11		First preload value.
12		Load step in which first preload value is to be applied.
13		Load step in which first preload value is to be locked.
14...		14 through 17 is a repeat of 10 through 13, but for the second preload value; 18 through 21 is for the third preload value; and so forth.

*GET Preprocessing Items, Entity = SECP

Entity* = SECP, *ENTNUM* = *NUM		
***GET, *Par*, SECP, *NUM*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
COUNT		Number of defined sections

Entity = SECP, *ENTNUM* = *NUM*		
***GET, *Par*, SECP, *NUM*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MAX		Largest section number defined
***Entity* = SECP, *ENTNUM* = *id* (beam section identification number)**		
***GET, *Par*, SECP, *id*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TYPE		Section type, for id - **SECTYPE** command. (always BEAM for beam sections)
SUBTYPE		Section type for id - **SECTYPE** command
NAME		Name defined for the given section id number
DATA	nnn	Where "nnn" is the location in the **SECDATA** command for the given section id number
PROP	AREA	Area value
"	IYY, IYZ, IZZ	Moments of inertia
"	WARP	Warping constant
"	TORS	Torsion constant
"	CGY, CGZ	Y or Z coordinate center of gravity
"	SHCY, SHCZ	Y or Z coordinate shear center
"	SCYY, SCYZ, SCZZ	Shear correction factors
"	OFFSET	Offset location: 1 = Centroid 2 = Shear Center 3 = Origin 0 = User Defined

*GET Preprocessing Items, Entity = SHEL

Entity = SHEL, ENTNUM = *N* (shell section identification number)		
***GET, Par, SHEL, N, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TYPE		Section type, for id — **SECTYPE** command. (always SHEL for shell sections)
NAME		Name defined for a given id number.
PROP	TTHK	Total thickness.
"	NLAY	Number of layers.
"	NSP	Number of section integration points.
"	POS	Node position (as defined by **SECOFFSET**).
"	"	0 = User Defined.
"	"	1 = Middle.
"	"	2 = Top.
"	"	3 = Bottom.
"	OFFZ	User-defined section offset (POS = 0).

Entity = SHEL, ENTNUM = *N* (shell section identification number)		
***GET, Par, SHEL, N, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
"	TS11	Transverse shear stiffness factors.
"	TS22	Transverse shear stiffness factors.
"	TS12	Transverse shear stiffness factors.
"	HORC	Homogeneous or complete section flag.
"	"	0 = Homogeneous.
"	"	1 = Composite.
"	FUNC	Tabular function name for total thickness.
"	UT11	User transverse shear stiffness 11.
"	UT22	User transverse shear stiffness 22.
"	UT12	User transverse shear stiffness 12.
"	AMAS	Added mass.
"	MSCF	Hourglass control membrane scale factor.
"	BSCF	Hourglass control bending scale factor.
"	DSTF	Drill stiffness scale factor.
"	LDEN	Laminate density.
LAYD	LayerNumber,THIC	Layer thickness.
"	LayerNumber,MAT	Layer material.
"	LayerNumber,ANGL	Layer orientation angle.
"	LayerNumber,NINT	Number of layer integration points.

*GET Preprocessing Items, Entity = TBFT

Entity = TBFT, *ENTNUM = BLANK*		
***GET, *Par*, TBFT, , Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
nmat		Number of defined material models.
matnum	*index*	Material number in array (index varies for 1 to num materials).
Entity = TBFT, *ENTNUM* = *matid* (For getting names of constitutive function, matid = the material ID number)		
***GET**, *Par*, TBFT, matid, nfun, IT1NUM, Item2, IT2NUM		
Item1	IT1NUM	Description
nfun		Number of constitutive functions for this material.
Entity = TBFT, *ENTNUM* = *matid* (To query constitutive function data, matid = the material ID number)		
***GET**, *Par*, TBFT, matid, func, fname, Item2, IT2NUM		
Item1	IT1NUM	Description
func	index	if Item2 = fname, the name of the constitutive function is returned.
func	function name	If Item2 = ncon, the number of constants is returned for the function specified in IT1NUM by the constitutive function name.
″	″	If Item2 = cons, set Item2num to index to return the value of the constant.
″	″	If Item2 = fixe, set Item2num to index to return the fix flag status.

Entity = TBFT, *ENTNUM* = *BLANK*		
***GET, *Par*, TBFT, , Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
"	"	If Item2 = type, returns the category of the constitutive model (moon, poly, etc.)
"	"	If Item2 = sord, returns the shear order of the prony visco model.
"	"	If Item2 = bord, returns the bulk order of the prony visco model.
"	"	If Item2 = shif, returns the shift function name of the prony visco model.
Entity = TBFT, *ENTNUM* = *matid* (To query experimental data, *matid* = the material ID number))		
***GET**, *Par*, TBFT, matid, func, fname, Item2, IT2NUM		
Item1	IT1NUM	Description
expe	(blank)	If Item2 = nexp, returns number of experiments in a material model.
"	expindex	If Item2 = type, returns index of experiment.
"	"	If Item2 = numrow, returns number of rows in the data.
"	"	If Item2 = numcol, returns the number of cols in a row (set Intem2num = Row index)
"	"	If Item2 = data, returns the value of the data in *row, col* of exp expindex (set item2Num = row index and item3 = column index. All indices vary from 1 to the maximum value.
"	"	If Item2 = natt, returns the number of attributes.
"	"	If Item2 = attname, returns the attribute name (set Item2Num = Attr index).
"	"	If Item2 = attvald, returns double value of attribute (set Item2Num = Attr index).
"	"	If Item2 = attvali, returns integer valud of attribute (set Item2Num = Attr index).
"	"	If Item2 = attvals, returns the string value of the attribute (set Item2Num = Attr index).

*GET Preprocessing Items, Entity = TBLAB

Entity = TBlab, *ENTNUM* = *N*..(*TBlab* = data table label from the TB command; *N* = material number.)		
***GET, *Par*, *TBlab*, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TEMP	*T*	**Item2**: CONST **IT2NUM**: *Num* Value of constant number Num in the datatable at temperature *T* (see Data Tables - Implicit Analysis in the *ANSYS Elements Reference*). For constants input a X,Y point, the constant numbers are consecutive with the X constants being the odd numbers, beginning with one.

*GET Preprocessing Items, Entity = VOLU

Entity = VOLU, *ENTNUM* = *N* (volume number)		
***GET, *Par*, VOLU, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ATTR	*Name*	Number assigned to the attribute *Name*, where *Name*=MAT, TYPE, REAL, ESYS, NNOD, or NELM. (NNOD=number of nodes, NELM=number of elements.)
VSEL		Select status of volume *N*: -1=unselected, 0=undefined, 1=selected. Alternative get function: VSEL(*N*).

Entity = VOLU, *ENTNUM* = *N* (volume number)		
***GET, *Par*, VOLU, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NXTH		Next higher volume number above *N* in selected set (or zero if none found). Alternative get function: VLNEXT(*N*).
NXTL		Next lower volume number below *N* in selected set (or zero if none found).
VOLU		Volume of volume *N*. (**VSUM** or **GSUM** must have been performed sometime previously with at least this volume *N* selected).
SHELL	1, 2, ..., *m*	**Item2**: AREA **IT2NUM**: 1,2,...,*p* Line number of position *p* of shell *m*
Entity = VOLU, *ENTNUM* = 0 (or blank)		
***GET**, *Par*, VOLU, 0, Item1, IT1NUM, Item2, IT2NUM		
NUM	MAX, MIN	Highest or lowest volume number in the selected set.
NUM	MAXD, MIND	Highest or lowest volume number defined.
COUNT		Number of volumes in the selected set.
VOLU		Combined volumes (from last **VSUM** or **GSUM**).
CENT	X, Y, Z	Centroid X, Y, or Z location of volumes (from last **VSUM** or **GSUM**).
IOR	X, Y, Z, XY, YZ, ZX	Moments of inertia about origin (from last **VSUM** or **GSUM**).
IMC	X, Y, Z, XY, YZ, ZX	Moments of inertia about mass centroid (from last **VSUM** or **GSUM**).
IPR	X, Y, Z	Principal moments of inertia (from last **VSUM** or **GSUM**).
IXV	X, Y, Z	Principal orientation X vector components (from last **VSUM** or **GSUM**).
IYV	X, Y, Z	Principal orientation Y vector components (from last **VSUM** or **GSUM**).
IZV	X, Y, Z	Principal orientation Z vector components (from last **VSUM** or **GSUM**).

*GET Preprocessing Items, Entity = WELD

Entity = WELD, *ENTNUM* = *N* (weld number)		
***GET, *Par*, WELD, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NODE	1, 2	First or second node number for spot weld *N*.
NXTH		Next higher spotweld number above *N* (or 0 if none found).
Entity = WELD, *ENTNUM* = 0 (or blank)		
***GET, *Par*, WELD, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NUM	MAX, MIN	Highest or lowest spotweld number.
COUNT		Total number of spotwelds in model.

Solution Items

*GET Solution Entity Items

- *GET Solution Items, Entity = ACTIVE

- *GET Solution Items, Entity = ELEM
- *GET Solution Items, Entity = MODE
- *GET Solution Items, Entity = RUNST

*GET Solution Items, Entity = ACTIVE

***Entity* = ACTIVE, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, ACTIVE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ANTY		Current analysis type.
SOLU	DTIME	Time step size.
"	NCMLS	Cumulative number of load steps.
"	NCMSS	Cumulative number of substeps. NOTE: Used only for static and transient analyses.
"	EQIT	Number of equilibrium iterations.
"	NCMIT	Cumulative number of iterations.
"	CNVG	Convergence indicator: 0=not converged, 1=converged.
"	MXDVL	Maximum degree of freedom value.
"	RESFRQ	Response frequency for 2nd order systems.
"	RESEIG	Response eigenvalue for 1st order systems.
"	DSPRM	Descent parameter.
"	FOCV	Force convergence value.
"	MOCV	Moment convergence value.
"	HFCV	Heat flow convergence value.
"	MFCV	Magnetic flux convergence value.
"	CSCV	Current segment convergence value.
"	CUCV	Current convergence value.
"	FFCV	Fluid flow convergence value.
"	DICV	Displacement convergence value.
"	ROCV	Rotation convergence value.
"	TECV	Temperature convergence value.
"	VMCV	Vector magnetic potential convergence value.
"	SMCV	Scalar magnetic potential convergence value.
"	VOCV	Voltage convergence value.
"	PRCV	Pressure convergence value.
"	VECV	Velocity convergence value.
"	CRPRAT	Maximum creep ratio.
"	PSINC	Maximum plastic strain increment.
"	CGITER	Number of iterations in the PCG and symmetric JCG (non-complex version) solvers.

*GET Solution Items, Entity = ELEM

***Entity* = ELEM, *ENTNUM* = 0 (or blank) (Available only after inertia relief solution [IRLF,1] or pre-calculation of masses [IRLF,-1])**		
***GET, *Par*, ELEM, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MTOT	X, Y, Z	Total mass components.
MC	X, Y, Z	Mass centroid components.
IOR	X, Y, Z, XY, YZ, ZX	Moment of inertia about origin.
IMC	X, Y, Z, XY, YZ, ZX	Moment of inertia about mass centroid.
FMC	X, Y, Z	Force components at mass centroid.
MMOR	X, Y, Z	Moment components at origin.
MMMC	X, Y, Z	Moment components at mass centroid.

*GET Solution Items, Entity = MODE

***Entity* = MODE, *ENTNUM* = *N* (mode number)**		
***GET, *Par*, MODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
FREQ	(IMAG)	Frequency of mode *N*. This item returns only the first 600 frequency values. For modal solutions that create complex frequencies (DAMP and QRDAMP), the real part of the frequencies is returned unless IT1NUM = IMAG.
PFACT		Participation factor of mode *N*. If retrieved after a modal analysis, this value reflects the participation factor for rotation about the global Z axis. If retrieved after a spectrum analysis, this value represents the participation factor for the last **SED** direction vector.
MCOEF		Mode coefficient of mode *N*. Values are retrievable with this command following a spectrum analysis. *Note — Note*--Values for the MCOEF parameter are only valid after a spectrum analysis has been solved.
DAMP		Effective damping ratio of mode *N*. Not a function of direction. Also retrievable following a Harmonic Response Analysis or Transient Analysis with mode superposition.

*GET Solution Items, Entity = RUNST

***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RSPEED command before retrieving the following items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
RSPEED	MIPS	MIPS rating of computer.
"	SMFLOP	Scalar MFLOPS rating of computer.
"	VMFLOP	Vector MFLOPS rating of computer.

***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RFILSZ command before retrieving these items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
RFILSZ	TOTAL	Estimated total size of all files listed in the **RFILSZ** command description. All file sizes are in megabytes.
"	EMAT	Estimated size of element matrices file (.EMAT).
"	EROT	Rotated element matrices file (.EROT).
"	ESAV	Element saved data file (.ESAV).
"	FULL	Assembled global stiffness and mass matrices file (.FULL).
"	MODE	Modal matrices file (.MODE).
"	RDSP	Reduced displacements file (.RDSP).
"	REDM	Reduced structure matrix file (.REDM).
"	RFRQ	Reduced complex displacements file (.RFRQ).
"	RGEOM	Geometry data in results file (.RST, .RTH, or .RMG).
"	RST	Load data in results file (.RST, .RTH, or .RMG).
"	TRI	Triangularized stiffness matrix file (.TRI).
Entity = RUNST, *ENTNUM* = 0 (or blank) Generate data using the **RTIMST** command before retrieving the following items:		
***GET**, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM		
Item1	IT1NUM	Description
RTIMST	TOTAL	Estimated run time (seconds) for total solution.
"	TFIRST	Estimated run time (seconds) for first iteration.
"	TITER	Estimated run time (seconds) for subsequent iteration.
"	EQPREP	Estimated run time (seconds) for element preparation.
"	SOLVE	Estimated run time (seconds) for wavefront solution.
"	BSUB	Estimated run time (seconds) for back substitution.
"	EIGEN	Estimated run time (seconds) for eigenvalue calculation.
"	ELFORM	*n* Estimated run time (seconds) for element formulation of element type number *n*.
"	ELSTRS	*n* Estimated run time (seconds) for computation of element results for element type number *n*.
"	NELM	*n* Number of elements defined for element type *n*.
Entity = RUNST, *ENTNUM* = 0 (or blank) Generate data using the **RMEMRY** command before retrieving the following items:		
***GET**, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM		
Item1	IT1NUM	Description
RMEMRY	WSREQ	Requested work space (Mb)
"	WSAVAIL	Work space obtained (Mb).
"	DBPSIZE	ANSYS database page size (Kb).
"	DBPDISK	Database pages on disk.
"	DBSIZE	ANSYS database space size (Mb).
"	DBPMEM	Database pages in memory.

***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RFILSZ command before retrieving these items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
"	DBMEM	Memory for database pages (Mb).
"	SCRSIZE	ANSYS scratch memory size (Mb).
"	SCRAVAIL	Available scratch memory (Mb).
"	IOMEM	Buffer scratch memory (Mb).
"	IOPSIZ	Binary I/O page size (Kb).
"	IOBUF	Buffers per solution file.
"	SOLMEMORY	Maximum Solution Memory Space
Entity = RUNST, *ENTNUM* = 0 (or blank) Generate data using the **RWFRNT** command before retrieving these items:		
***GET**, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM		
Item1	IT1NUM	Description
RWFRNT	MAX	Estimated maximum wavefront.
"	RMS	Estimated R.M.S. wavefront.
"	MEAN	Estimated mean wavefront.
***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RTIMST command before retrieving the following items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
RTIMST	TOTAL	Estimated run time (seconds) for total solution.
"	TFIRST	Estimated run time (seconds) for first iteration.
"	TITER	Estimated run time (seconds) for subsequent iteration.
"	EQPREP	Estimated run time (seconds) for element preparation.
"	SOLVE	Estimated run time (seconds) for wavefront solution.
"	BSUB	Estimated run time (seconds) for back substitution.
"	EIGEN	Estimated run time (seconds) for eigenvalue calculation.
"	ELFORM	*n* Estimated run time (seconds) for element formulation of element type number *n*.
"	ELSTRS	*n* Estimated run time (seconds) for computation of element results for element type number *n*.
"	NELM	*n* Number of elements defined for element type *n*.
***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RMEMRY command before retrieving the following items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
RMEMRY	WSREQ	Requested work space (Mb)
"	WSAVAIL	Work space obtained (Mb).
"	DBPSIZE	ANSYS database page size (Kb).
"	DBPDISK	Database pages on disk.
"	DBSIZE	ANSYS database space size (Mb).

***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RMEMRY command before retrieving the following items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
"	DBPMEM	Database pages in memory.
"	DBMEM	Memory for database pages (Mb).
"	SCRSIZE	ANSYS scratch memory size (Mb).
"	SCRAVAIL	Available scratch memory (Mb).
"	IOMEM	Buffer scratch memory (Mb).
"	IOPSIZ	Binary I/O page size (Kb).
"	IOBUF	Buffers per solution file.
"	SOLMEMORY	Maximum Solution Memory Space
***Entity* = RUNST, *ENTNUM* = 0 (or blank) Generate data using the RWFRNT command before retrieving these items:**		
***GET, *Par*, RUNST, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
RWFRNT	MAX	Estimated maximum wavefront.
"	RMS	Estimated R.M.S. wavefront.
"	MEAN	Estimated mean wavefront.

Postprocessing Items

*GET Postprocessing Entity Items

- *GET Postprocessing Items, Entity = ACTIVE
- *GET Postprocessing Items, Entity = ELEM
- *GET Postprocessing Items, Entity = ETAB
- *GET Postprocessing Items, Entity = FSUM
- *GET Postprocessing Items, Entity = INTSRF
- *GET Postprocessing Items, Entity = KCALC
- *GET Postprocessing Items, Entity = NODE
- *GET Postprocessing Items, Entity = PATH
- *GET Postprocessing Items, Entity = PLNSOL
- *GET Postprocessing Items, Entity = PRERR
- *GET Postprocessing Items, Entity = RAD
- *GET Postprocessing Items, Entity = SECR
- *GET Postprocessing Items, Entity = SECTION
- *GET Postprocessing Items, Entity = SORT
- *GET Postprocessing Items, Entity = SSUM
- *GET Postprocessing Items, Entity = TREF

- *GET Postprocessing Items, Entity = VARI

*GET Postprocessing Items, Entity = ACTIVE

***Entity* = ACTIVE, *ENTNUM* = 0 (or blank)**		
***GET,*Par*, ACTIVE, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SET	LSTP	Current load step number.
"	SBST	Current substep number.
"	TIME	Time associated with current results in the database.
"	FREQ	Frequency (for ANTYPE=MODAL, HARMIC, SPECTR; load factor for ANTYPE=BUCKLE).
"	NSET	If Item2 is blank, number of data sets on result file. If Item2 = FIRST, IT2NUM = Loadstep, get set number of first iteration of Loadstep If Item2 = LAST, IT2NUM = Loadstep, get set number of last iteration of Loadstep
RSYS		Active results coordinate system.

*GET Postprocessing Items, Entity = ELEM

***Entity* = ELEM, *ENTNUM* = *N* (element number)**		
***GET,*Par*, ELEM, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SERR		Structural error energy.
SDSG		Absolute value of the maximum variation of any nodal stress component.
TERR		Thermal error energy.
TDSG		Absolute value of the maximum variation of any nodal thermal gradient component.
SENE		"Stiffness" energy or thermal heat dissipation. Same as TENE.
TENE		Thermal heat dissipation or "stiffness" energy. Same as SENE.
KENE		Kinetic energy.
JHEAT		Element Joule heat generation (coupled-field calculation).
JS	X, Y, Z	Source current density (coupled-field calculation) in the global Cartesian coordinate system.
HS	X, Y, Z	Average element magnetic field intensity from current sources.
VOLU		Element volume, as calculated during solution.
ETAB	*Lab*	Value of element table item *Lab* for element *N* (see **ETABLE** command).
SMISC	*Snum*	Value of element summable miscellaneous data at sequence number *Snum* (as used on **ETABLE** command).
NMISC	*Snum*	Value of element non-summable miscellaneous data at sequence number *Snum* (as used on **ETABLE** command).

*GET Postprocessing Items, Entity = ETAB

***Entity* = ETAB, *ENTNUM* = *N* (column number)**		
***GET,*Par*, ETAB, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
LAB		Label for column *N* of the element table [**ETABLE**]. Returns a character parameter.
ELEM	*E*	Value in **ETABLE** column *N* for element number *E*.
***Entity* = ETAB, *ENTNUM* = 0 (or blank)**		
***GET,*Par*,ETAB,0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NCOL	MAX	Total number of **ETABLE** columns.
NLENG	MAX	Largest element number defined.

*GET Postprocessing Items, Entity = FSUM

***Entity* = FSUM, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, FSUM, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ITEM	*Lab*	Value of item *Lab* from last **FSUM** command. Valid labels are FX, FY, FZ, MX, MY, MZ, FLOW, HEAT, FLUX, etc.

*GET Postprocessing Items, Entity = INTSRF

***Entity* = INTSRF, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, INTSRF, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
PRES	*Lab*	Value of item *Lab* from last **INTSRF**,PRES command. Valid labels are FX, FY, FZ, MX, MY, and MZ.
TAUW	*Lab*	Value of item *Lab* from last **INTSRF**,TAUW command. Valid labels are FX, FY, FZ, MX, MY, and MZ.

*GET Postprocessing Items, Entity = KCALC

***Entity* = KCALC, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, KCALC, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
K	1, 2, 3	Value of KI, KII, or KIII stress intensity factor from last **KCALC** command.

*GET Postprocessing Items, Entity = NODE

***Entity* = *NODE*, *ENTNUM* = *N* (node number) for nodal degree of freedom results:**		
***GET, *Par*, NODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
U	X, Y, Z, SUM	X, Y, or Z structural displacement or vector sum. Alternative get functions: UX(*N*), UY(*N*), UZ(*N*).
ROT	X, Y, Z, SUM	X, Y, or Z structural rotation or vector sum. Alternative get functions: ROTX(*N*), ROTY(*N*), ROTZ(*N*).
TEMP		Temperature. For **SHELL131** and **SHELL132** elements with KEYOPT(3) = 0 or 1, use TBOT, TE2, TE3, . . ., TTOP instead of TEMP. Alternative get functions: TEMP(*N*), TBOT(*N*), TE2(*N*), etc.
PRES		Pressure. Alternative get function: PRES(*N*).
VOLT		Electric potential. Alternative get function: VOLT(*N*).
MAG		Magnetic scalar potential. Alternative get function: MAG(*N*).
V	X, Y, Z, SUM	X, Y, or Z fluid velocity or vector sum in a fluid analysis. X, Y, or Z nodal velocity or vector sum in a structural transient analysis (LS-DYNA analysis or ANSYS analysis with **ANTYPE**,TRANS). Alternative get functions: VX(*N*), VY(*N*), VZ(*N*).
A	X, Y, Z, SUM	X, Y, or Z magnetic vector potential or vector sum in an electromagnetic analysis. X, Y, or Z nodal acceleration or vector sum in a structural transient analysis (LS-DYNA analysis or ANSYS analysis with **ANTYPE**,TRANS). Alternative get functions: AX(*N*), AY(*N*), AZ(*N*).
CURR		Current.
EMF		Electromotive force drop.
ENKE		Turbulent kinetic energy (FLOTRAN). Alternative get function: ENKE(*N*).
ENDS		Turbulent energy dissipation (FLOTRAN). Alternative get function: ENDS(*N*).
RF	FX, FY, FZ, MX, MY, MZ	Nodal reaction forces in the nodal coordinate system.
Note: This command should be used very carefully when N stands for an internal node, since the nodal degree of freedoms may have different physical meanings.		
***Entity* = *NODE*, *ENTNUM* = *N* (node number) for element nodal results:**		
***GET, *Par*, NODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
S	X, Y, Z, XY, YZ, XZ	Component stress.
"	1, 2, 3	Principal stress.
"	INT, EQV	Stress intensity or equivalent stress.
"	MAXF	Maximum stress failure criterion.
"	TWSI	Tsai-Wu strength failure criterion.
"	TWSR	Inverse of Tsai-Wu strength ratio index failure criterion.
EPTO	X, Y, Z, XY, YZ, XZ,	Component total strain (EPEL + EPPL + EPCR).
"	1, 2, 3	Principal total strain.
"	INT, EQV	Total strain intensity or total equivalent strain.

***Entity* = *NODE*, *ENTNUM* = *N* (node number) for element nodal results:**		
***GET, *Par*, NODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
EPEL	X, Y, Z, XY, YZ, XZ	Component elastic strain.
"	1, 2, 3	Principal elastic strain.
"	INT, EQV	Elastic strain intensity or elastic equivalent strain.
"	MAXF	Maximum strain failure criterion.
EPPL	X, Y, Z, XY, YZ, XZ	Component plastic strain.
"	1, 2, 3	Principal plastic strain.
"	INT, EQV	Plastic strain intensity or plastic equivalent strain.
EPCR	X, Y, Z, XY, YZ, XZ	Component creep strain.
"	1, 2, 3	Principal creep strain.
"	INT, EQV	Creep strain intensity or creep equivalent strain.
GKS	X, XY, XZ	Gasket component stress.
GKD	X, XY, XZ	Gasket component total closure.
GKDI	X, XY, XZ	Gasket component total inelastic closure.
GKTH	X, XY, XZ	Gasket component thermal closure.
EPTH	X, Y, Z, XY, YZ, XZ	Component thermal strain.
"	1, 2, 3	Principal thermal strain.
"	INT, EQV	Thermal strain intensity or thermal equivalent strain.
EPSW		Swelling strain.
NL	SEPL	Equivalent stress (from stress-strain curve).
"	SRAT	Stress state ratio.
"	HPRES	Hydrostatic pressure.
"	EPEQ	Accumulated equivalent plastic strain.
"	PSV	Plastic state variable or plastic work/volume.
"	PLWK	Plastic work/volume.
TG	X, Y, Z, SUM	Component thermal gradient and sum.
TF	X, Y, Z, SUM	Component thermal flux and sum.
PG	X, Y, Z, SUM	Component pressure gradient and sum.
EF	X, Y, Z, SUM	Component electric field and sum.
D	X, Y, Z, SUM	Component electric flux density and sum.
H	X, Y, Z, SUM	Component magnetic field intensity and sum.
B	X, Y, Z, SUM	Component magnetic flux density and sum.
FMAG	X, Y, Z, SUM	Component magnetic force and sum.
HS	X, Y, Z	Component magnetic field intensity from current sources (in the global Cartesian coordinate system).
BFE	TEMP	Body temperatures (calculated from applied temperatures) as used in solution.

***Entity* = *NODE*, *ENTNUM* = *N* (node number) for FLOTRAN results:**		
***GET, *Par*, NODE, *N*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TTOT		Total temperature.
HFLU		Heat flux.
HFLM		Heat transfer (film) coefficient.
COND		Fluid laminar conductivity.
PCOE		Pressure coefficient.
PTOT		Total (stagnation) pressure.
MACH		Mach number.
STRM		Stream function (2-D applications only).
DENS		Fluid density.
VISC		Fluid laminar viscosity.
EVIS		Fluid effective viscosity.
CMUV		Turbulent viscosity coefficient.
ECON		Fluid effective conductivity.
YPLU		Y+, a turbulent law of the wall parameter.
TAUW		Shear stress at the wall.

*GET Postprocessing Items, Entity = PATH

Entity = PATH, *ENTNUM* = 0 (or blank)		
***GET, *Par*, PATH, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MAX	*Lab*	Maximum value of path item *Lab* from last path operation. Valid labels are the user-defined labels on the **PDEF** or **PCALC** command.
MAXPATH		Returns the maximum path number defined.
MIN	*Lab*	Minimum value of path item *Lab* from last path operation. Valid labels are the user-defined labels on the **PDEF** or **PCALC** command.
LAST	*Lab*	Last value of path item *Lab* from last path operation. Valid labels are the user-defined labels on the **PDEF** or **PCALC** command.
NODE		Value providing the number of nodes defining the path referenced in the last path operation.
ITEM	*Lab*	***Item2*** = PATHPT, ***IT2NUM*** = *n* The value of *Lab* at the *n*th data point from the last path operation.
POINT	*n*	***Item2*** = *X,Y,Z*, or *CSYS*. Returns information about the nth point on the current path.
NVAL		The number of path data points (the length of the data table) from the last path operation.
SET	*n*	***Item2*** = NAME. Returns the name of the *n*th data set on the current path.
NUMPATH		Returns the number of paths defined.

***Entity* = PATH, *ENTNUM* = n (path number)**		
Item1	**IT1NUM**	**Description**
NAME		Returns the name of the *n*th path.
***Entity* = PATH, *ENTNUM* = 0 (or blank)**		
***GET,*Par*,KCALC,0,...**		
K	1, 2, 3	Value of KI, KII, or KIII stress intensity factor from last **KCALC** command.

*GET Postprocessing Items, Entity = PLNSOL

***Entity* = PLNSOL, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, PLNSOL, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MAX		Maximum value of item in last contour display [**PLNSOL**].
MIN		Minimum value of item in last contour display [**PLNSOL**].
BMAX		Maximum bound value of item in last contour display [**PLNSOL**].
BMIN		Minimum bound value of item in last contour display [**PLNSOL**].

*GET Postprocessing Items, Entity = PRERR

***Entity* = PRERR, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, PRERR, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
SEPC		Structural percent error in energy norm [**PRERR**].
TEPC		Thermal percent error in energy norm [**PRERR**].
SERSM		Structural error energy summation [**PRERR**].
TERSM		Thermal error energy summation [**PRERR**].
SENSM		Structural energy summation [**PRERR**].
TENSM		Thermal energy summation [**PRERR**].

*GET Postprocessing Items, Entity = RAD

***Entity* = RAD, ENTNUM = 0 (or blank)**		
***GET, *Par*, RAD, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
VFAVG		Value of the average view factor computed from the previous VFQUERY command.
***Entity* = RAD, ENTNUM = n (enclosure number)**		
***GET, *Par*, RAD, n**		
Item1	**IT1NUM**	**Description**
NETHF		Value of the net heat rate lost by an enclosure.

*GET Postprocessing Items, Entity = SECR

***Entity* = SECR, *ENTNUM* = *n* (element number) If the element number (n) is blank, or ALL, find the max or min of all the elements.**		
***GET, *Par*, SECR, n, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
S	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return highest or lowest component total stress.
EPTO	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return the highest or lowest component total strain
EPTH	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return the highest or lowest component thermal strain
EPPL	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return the highest or lowest component plastic strain
PLWK	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return the highest or lowest nonlinear item plastic work
EPCR	X, XZ, XY	Item 2 = MAX, or Item2 =MIN Return the highest or lowest component creep strain

*GET Postprocessing Items, Entity = SECTION

***Entity* = SECTION,*ENTNUM* = *component* (listed below).** **Generate data for section stress results, using PRSECT or PLSECT, before retrieving these items. Valid labels for *ENTNUM* are MEMBRANE, BENDING, SUM (Membrane+Bending) , PEAK, and TOTAL. (The following items are not stored in the database and the values returned reflect the last quantities generated by PRSECT or PLSECT.) Only MEMBRANE, BENDING, and SUM data are available after a PLSECT command.**		
***GET, *Par*, SECTION, *component*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
INSIDE	SX, SY, SZ, SXY, SYZ, SXZ	Stress component at beginning of path.
CENTER	SX, SY, SZ, SXY, SYZ, SXZ	Stress component at midpoint of path.
OUTSIDE	SX, SY, SZ, SXY, SYZ, SXZ	Stress component at end of path.

*GET Postprocessing Items, Entity = SORT

***Entity* = SORT, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, SORT, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
MAX		Maximum value of last sorted item (**NSORT** or **ESORT** command).
MIN		Minimum value of last sorted item (**NSORT** or **ESORT** command).
IMAX		Node/Element number where maximum value occurs.
IMIN		Node/Element number where minimum value occurs.

*GET Postprocessing Items, Entity = SSUM

Entity = SSUM, *ENTNUM* = 0 (or blank)		
*GET, *Par*, SSUM, 0, Item1, IT1NUM, Item2, IT2NUM		
Item1	**IT1NUM**	**Description**
ITEM	*Lab*	Value of item *Lab* from last **SSUM** command. Valid labels are the user-defined labels on the **ETABLE** command.

*GET Postprocessing Items, Entity = TREF

Entity = TREF, *ENTNUM* = 0 (or blank)		
*GET, *Par*, TREF, 0, Item1, IT1NUM, Item2, IT2NUM		
Item1	**IT1NUM**	**Description**
ENER		Stored energy.
ENUM		Trefftz element number.
CEMIN		First (or minimum) constraint equation number associated with the Trefftz domain.
CEMAX		Last (or maximum) constraint equation number associated with the Trefftz domain.
NTZN		Number of Trefftz DOFs.
NSFN		Number of nodes on exterior surface.
NSFE		Number of faces on exterior element.

*GET Postprocessing Items, Entity = VARI

Entity = VARI, *ENTNUM* = *N* (variable number after POST26 data storage) (for complex values, only the real portion is returned)		
*GET,*Par, VARI, N*, Item1, IT1NUM, Item2, IT2NUM		
Item1	**IT1NUM**	**Description**
EXTREM	VMAX	Maximum extreme value
"	TMAX	Time corresponding to VMAX.
"	VMIN	Minimum extreme value (after POST26 data storage).
"	TMIN	Time corresponding to VMIN.
"	VLAST	Last value (after POST26 data storage).
"	TLAST	Time corresponding to VLAST.
"	CVAR	Covariance
RTIME	t	Real value of variable *N* at time=*t*.
ITIME	t	Imaginary value of variable *N* at time=*t*.
RSET	*Snum*	Real value of variable *N* at location *Snum*.
ISET	*Snum*	Imaginary value of variable *N* at location *Snum*.
Entity = VARI, *ENTNUM* = 0 (or blank) (after POST26 data storage)		
*GET,*Par*,VARI,0, Item1, IT1NUM, Item2, IT2NUM		
Item1	**IT1NUM**	**Description**
NSETS		Number of data sets stored (after POST26 data storage).

Optimization and Probabilistic Design

*GET Optimization and Probabilistic Design Entity Items

- *GET Optimization and Probabilistic Design Items, Entity = OPT
- *GET Optimization and Probabilistic Design Items, Entity = TOPO
- *GET Optimization and Probabilistic Design Items, Entity = PDS (pre)
- *GET Optimization and Probabilistic Design Items, Entity = PDS (post)

*GET Optimization and Probabilistic Design Items, Entity = OPT

***Entity* = OPT, *ENTNUM* = 0 (or blank)**		
***GET, *Par*, OPT, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
TOTAL		Total number of analysis loops that have been executed.
ITER		Total number of iterations for the optimization method or tool (i.e., per **OPEXE** command).
FEAS	*N*	Feasibility of design set *N:* 0=infeasible, 1=feasible.
TERM		Termination condition. For first order [**OPTYPE**,FIRST] or subproblem approximation [**OPTYPE**,SUBP] optimization: -1=not converged or not finished (still looping), 0=converged, 1=not converged due to too many sequential infeasible designs; 2=not converged due to too many iterations. For all other optimization methods: -1=not finished (still looping), 3=optimization complete.
BEST		Best design set. If design is feasible, best design is the one with the lowest value of the objective function. If infeasible, the best design is the one that is closest to being feasible.

*GET Optimization and Probabilistic Design Items, Entity = TOPO

***Entity* = TOPO, ENTNUM = 0 (or blank)**		
***GET, *Par*, TOPO, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
ACT		Status of topological optimization: 0=off, 1=on
TOELEM		Total number of elements used for topological optimization.
LOADS		Number of load cases specified in the **TOCOMP** command.
ITER		Current number of iterations performed. The iteration counter retrieved is always one greater than the actual iterations performed because the densities are also one iteration ahead (more current than those you are seeing).
MXIT		Maximum number of topological optimization iterations allowed.
CONV		Termination/convergence indicator: 0=not converged, 1=converged.
DIM		Dimensionality of the topological optimization problem: 0=2D, 1=3D, 2=Shell.
KAXI		Axisymmetric option within 2-D topological optimization: 0=off, 1=on.
POWP		Power of power law within axisymmetric option.
NEV		Total number of eigenvalues considered in topological optimization.
TOAC		Termination/convergence accuracy.

***Entity* = TOPO, ENTNUM = 0 (or blank)**		
***GET, *Par*, TOPO, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
LOWD		Lower bound for element densities.
NTOC		Total number of constraints defined for topological optimization problem.
SFLAG		Solution approach active: 0=OC, 1=SCP.
COMP		Compliance value for current iteration.
PORV		Porous volume value for current iteration.
Entity* = TOPO, *ENTNUM* = *n		
***GET, *Par*, TOPO, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
DENS		Element density used for topological optimization: low value (near 0)=material to be removed, high value (near 1)=material to keep.*n* = element number
FRQI		Individual frequencies for current topological optimization iteration. *n* =frequency ID
TCBO	FLAG	Constraint bounds for topological optimization: FLAG=1 - Lower bound. FLAG=2 - Upper bound. *n* = constraint ID
TCBF		Indicate whether bound should be treated as percentage of initial design or as an absolute value: 0=percentage, 1=absolute.*n* = constraint ID
TOHO		Value of topological objective for specified iteration.*n* = iteration counter
TOHC	CID	Value of topological constraint CID for specified iteration.*n* = iteration counter

*GET Optimization and Probabilistic Design Items, Entity = PDS (pre)

***Entity* = PDS, ENTNUM = 0 (or blank)**		
***GET, *Par*, PDS, 0, Item1, IT1NUM, Item2, IT2NUM.**		
Item1	**IT1NUM**	**Description**
ANLN		*Item2* = START; *IT2NUM* = *N*; Name of the analysis file containing the deterministic model. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all 32 characters.
ANLX		Extension of the analysis file containing the deterministic model. A character parameter of up to 8 characters is returned.
ANLD		*Item2* = START; *IT2NUM* = *N*; Name of the directory of the analysis file containing the deterministic model. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all 64 characters.
ASTP	1	Current setting for Autostop option (0 = disabled, 1 = enabled).
"	2	Current Autostop mean value accuracy.
"	3	Current Autostop standard deviation accuracy.
"	4	Current Autostop convergence checking frequency.
CORR	*i*	*Item2* = 0 or blank; *IT2NUM* = *j*; Correlation coefficient specified by the user between the *i*-th and the *j*-th random input variable. If no correlation has been specified between these two random variables a value of 0.0 is returned.
METH		Name of the current analysis method as specified in the **PDMETH** command (MCS or RSM). A character parameter of up to 8 characters is returned.

***Entity* = PDS, ENTNUM = 0 (or blank)**		
***GET, *Par*, PDS, 0, Item1, IT1NUM, Item2, IT2NUM.**		
Item1	**IT1NUM**	**Description**
NSIM		Number of simulation loops requested, as specified in the **PDMETH** and **PDDMCS**, **PDLHS**, **PDDOEL**, or **PDUSER** commands.
NTRP		Current total number of defined random output parameters.
NTRV		Current total number of defined random input variables
PAR1	*i*	First distribution parameter of the *i*-th defined random input variable.
PAR2	*i*	Second distribution parameter of the *i*-th defined random input variable.
PAR3	*i*	Third distribution parameter of the *i*-th defined random input variable.
PAR4	*i*	Fourth distribution parameter of the *i*-th defined random input variable.
RNAM	*i*	*Item2* = START; *IT2NUM* = *N*; Name of the *i*-th defined random output parameter. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all characters.
SAMP		Name of the current sampling technique as specified in the **PDMETH** command (LHS, DIR, USER for MCS, or CCD, BBM, USER for RSM). A character parameter of up to 8 characters is returned.
VDIS	*i*	Label of the distribution type of the *i*-th defined random input variable (BETA, EXPO, ..., WEIB). A character parameter of up to 8 characters is returned.
VNAM	*i*	*Item2* = START; *IT2NUM* = *N*; Name of the *i*-th defined random input variable. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all characters.
CCDL		*Item2* = DEFA; *IT2NUM* = *j*; Default value for the probabilities of the *j*-th design-of-experiment level for a central composite design as used in the by the **PDDOEL**command.
CCDL	*i*	*Item2* = VTYP; *IT2NUM=0 (or blank)*; Type of the level values of the design-of-experiment for a central composite design of the *i*-th defined random input variable as specified by the **PDDOEL**command (PROB, PHYS). A character parameter of up to 8 characters is returned.
CCDL	*i*	*Item2* = LOPT; *IT2NUM=0 (or blank)*; Type of the level definition of the design-of-experiment for a central composite design of the *i*-th defined random input variable as specified by the **PDDOEL**command (BND, ALL). A character parameter of up to 8 characters is returned.
CCDL	*i*	*Item2* = LDEF; *IT2NUM=j*; Flag indicating if the *j*-th design-of-experiment level for a central composite design of the *i*-th defined random input variable has been defined with the **PDDOEL**command. (0=NO, 1=YES).
CCDL	*i*	*Item2* = LVAL; *IT2NUM=j*; Level value for the *j*-th design-of-experiment level for a central composite design of the *i*-th defined random input variable as specified by the **PDDOEL**command. If the **PDDOEL**command has not been used for the *i*-th defined random input variable or if the user has not specified the *j*-th design-of-experiment level, then the default probability level will be returned.
BBML		*Item2* = DEFA; *IT2NUM* = *j*; Default value for the probabilities of the *j*-th design-of-experiment level for a Box-Behnken Matrix design as used in the by the **PDDOEL**command.
BBML	*i*	*Item2* = VTYP; *IT2NUM=0 (or blank)*; Type of the level values of the design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable as specified by the **PDDOEL**command. (PROB, PHYS). A character parameter of up to 8 characters is returned.

Entity = PDS, ENTNUM = 0 (or blank)		
***GET, *Par*, PDS, 0, Item1, IT1NUM, Item2, IT2NUM.**		
Item1	**IT1NUM**	**Description**
BBML	*i*	*Item2* = LOPT; *IT2NUM=0 (or blank)*; Type of the level definition of the design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable as specified by the **PDDOEL**command (BND, ALL). A character parameter of up to 8 characters is returned.
BBML	*i*	*Item2* = LDEF; *IT2NUM=j*; Flag indicating if the *j*-th design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable has been defined with the **PDDOEL**command. (0=NO, 1=YES).
BBML	*i*	*Item2* = LVAL; *IT2NUM=j*; Level value for the *j*-th design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable as specified by the **PDDOEL**command. If the **PDDOEL**command has not been used for the *i*-th defined random input variable or if the user has not specified the *j*-th design-of-experiment level, then the default probability level will be returned.

*GET Optimization and Probabilistic Design Items, Entity = PDS (post)

Entity = PDS, ENTNUM = 0 (or blank)		
***GET, *Par*, PDS, 0, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
NRSS		Number of response surface sets that are currently available.
NSOL		Number of probabilistic solution sets that are currently available. This coincides with the number of probabilistic analyses that are currently done.
***Entity* = PDS, ENTNUM = *n* (*n*-th result set).** **The numbering or ordering of the result sets is used as follows: If NSOL is the number of solution sets and NRSS is the number response surface sets then the solution sets are indexed from 1 to NSOL and the response surface sets are indexed from NSOL+1 to NSOL+NRSS. Note that some options listed below apply only to solution sets (i.e., where $1 \le n \le$ NSOL) and some apply only to response surface sets (i.e., where NSOL+1 $\le n \le$ NSOL+NRSS).**		
***GET,*Par*, PDS, *n*, Item1, IT1NUM, Item2, IT2NUM**		
Item1	**IT1NUM**	**Description**
METH		Name of the analysis method used in the *n*-th solution set (MCS, RSM). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., "*n*" ranges from 1 to NSOL).
NSIM		Number of simulation samples that are available for postprocessing in the *n*-th result set. If the *n*-th result set is a solution set (i.e., $1 \le n \le$ NSOL) then this is equal to the number of successful (no error occurred) finite element analysis loops regardless of the probabilistic method used for the solution set. If *n* points to a response surface set, then this is equal to the number of simulation samples that were performed on the response surfaces included in this response surface set.
SAMP		Name of the sampling technique used in the *n*-th solution set (LHS, DIR, USER for MCS or CCD, BBM, USER for RSM). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).

***Entity* = PDS, ENTNUM = *n* (*n*-th result set).**

The numbering or ordering of the result sets is used as follows: If NSOL is the number of solution sets and NRSS is the number response surface sets then the solution sets are indexed from 1 to NSOL and the response surface sets are indexed from NSOL+1 to NSOL+NRSS. Note that some options listed below apply only to solution sets (i.e., where 1 $\leq$ *n* $\leq$ NSOL) and some apply only to response surface sets (i.e., where NSOL+1 $\leq$ *n* $\leq$ NSOL+NRSS).

***GET,*Par*, PDS, *n*, Item1, IT1NUM, Item2, IT2NUM**

Item1	IT1NUM	Description
RLAB		*Item2* = START; *IT2NUM* = *N*; Name of the *n*-th defined result set. A character parameter of up to 8 characters, starting at position *N*, is returned. Use ***DIM** and ***DO** to get all characters.
MEAN		*Item2* = RV; *IT2NUM=j*; The mean value of the *j*-th defined random input variable in the *n*-th result set.
MEAN		*Item2* = RP; *IT2NUM=j*; The mean value of the *j*-th defined random output parameter in the *n*-th result set.
STDV		*Item2* = RV; *IT2NUM=j*; The standard deviation of the *j*-th defined random input variable in the *n*-th result set.
STDV		*Item2* = RP; *IT2NUM=j*; The standard deviation of the *j*-th defined random output parameter in the *n*-th result set.
KURT		*Item2* = RV; *IT2NUM=j*; The coefficient of kurtosis of the *j*-th defined random input variable in the *n*-th result set.
KURT		*Item2* = RP; *IT2NUM=j*; The coefficient of kurtosis of the *j*-th defined random output parameter in the *n*-th result set.
SKEW		*Item2* = RV; *IT2NUM=j*; The coefficient of skewness of the *j*-th defined random input variable in the *n*-th result set.
SKEW		*Item2* = RP; *IT2NUM=j*; The coefficient of skewness of the *j*-th defined random output parameter in the *n*-th result set.
MIN		*Item2* = RV; *IT2NUM=j*; The minimum sampled value of the *j*-th defined random input variable in the *n*-th result set.
MIN		*Item2* = RP; *IT2NUM=j*; The minimum sampled value of the *j*-th defined random output parameter in the *n*-th result set.
MAX		*Item2* = RV; *IT2NUM=j*; The minimum sampled value of the *j*-th defined random input variable in the *n*-th result set.
MAX		*Item2* = RP; *IT2NUM=j*; The minimum sampled value of the *j*-th defined random output parameter in the *n*-th result set.
CCDL	*i*	*Item2* = VTYP; Type of the level values of the design-of-experiment for a central composite design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set (PROB, PHYS). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
CCDL	*i*	*Item2* = LOPT; Type of the level definition of the design-of-experiment for a central composite design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set (BND, ALL). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).

***Entity* = PDS, ENTNUM = *n* (*n*-th result set).**

The numbering or ordering of the result sets is used as follows: If NSOL is the number of solution sets and NRSS is the number response surface sets then the solution sets are indexed from 1 to NSOL and the response surface sets are indexed from NSOL+1 to NSOL+NRSS. Note that some options listed below apply only to solution sets (i.e., where 1 $\leq$ *n* $\leq$ NSOL) and some apply only to response surface sets (i.e., where NSOL+1 $\leq$ *n* $\leq$ NSOL+NRSS).

***GET,*Par*, PDS, *n*, Item1, IT1NUM, Item2, IT2NUM**

Item1	IT1NUM	Description
CCDL	*i*	*Item2* = LDEF; *IT2NUM=j*; Flag indicating if the *j*-th design-of-experiment level for a central composite design of the *i*-th defined random input variable has been specified for the *n*-th solution set. (0=NO, 1=YES). This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
CCDL	*i*	*Item2* = LVAL; *IT2NUM=j*; Level value for the *j*-th design-of-experiment level for a central composite design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
BBML	*i*	*Item2* = VTYP; Type of the level values of the design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set. (PROB, PHYS). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
BBML	*i*	*Item2* = LOPT; Type of the level definition of the design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set (BND, ALL). A character parameter of up to 8 characters is returned. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
BBML	*i*	*Item2* = LDEF; *IT2NUM=j*; Flag indicating if the *j*-th design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable specified for the *n*-th solution set. (0=NO, 1=YES).
BBML	*i*	*Item2* = LVAL; *IT2NUM=j*; Level value for the *j*-th design-of-experiment level for a Box-Behnken Matrix design of the *i*-th defined random input variable that has been used during the execution of the *n*-th solution set. This applies only to solution sets (i.e., *n* ranges from 1 to NSOL).
RSST		*Item2* = XSOL; Index of the solution set the response surface set identified by the *n*-th result set is associated with. This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSST		*Item2* = NFRP; Number of fitted random output parameters in the response surface set identified by the *n*-th result set is associated with. This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSUR	*i*	*Item2* = XFRP; Index of the random output parameter that has been fitted with the **RSFIT**command to fit the *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSUR	*i*	*Item2* = RMOD; Label for the response surface model that has been used in the **RSFIT**command to fit the *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). A character parameter of up to 8 characters is returned. This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).

***Entity* = PDS, ENTNUM = *n* (*n*-th result set).**

The numbering or ordering of the result sets is used as follows: If NSOL is the number of solution sets and NRSS is the number response surface sets then the solution sets are indexed from 1 to NSOL and the response surface sets are indexed from NSOL+1 to NSOL+NRSS. Note that some options listed below apply only to solution sets (i.e., where 1 $\leq$ *n* $\leq$ NSOL) and some apply only to response surface sets (i.e., where NSOL+1 $\leq$ *n* $\leq$ NSOL+NRSS).

***GET,*Par*, PDS, *n*, Item1, IT1NUM, Item2, IT2NUM**

Item1	IT1NUM	Description
RSUR	*i*	*Item2* = YTRT; Label of the type of transformation that has been used in the **RSFIT**command to fit the random output parameter of *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). A character parameter of up to 8 characters is returned. This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSUR	*i*	*Item2* = YTRV; Transformation value of the transformation that has been used in the **RSFIT**command to fit the random output parameter of *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSUR	*i*	*Item2* = FILT; Label of the filtering type of the regression terms that has been used in the **RSFIT**command to fit the random output parameter of *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). A character parameter of up to 8 characters is returned. This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSUR	*i*	*Item2* = CONF; Confidence level value of the regression term filtering that has been used in the **RSFIT**command to fit the random output parameter of *i*-th response surface in the response surface set identified by the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSEQ	*i*	*Item2* = YBOX; Box-Cox transformation value "lambda" of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSEQ	*i*	*Item2* = NTRM; Number of regression terms of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).

***Entity* = PDS, ENTNUM = *n* (*n*-th result set).**

The numbering or ordering of the result sets is used as follows: If NSOL is the number of solution sets and NRSS is the number response surface sets then the solution sets are indexed from 1 to NSOL and the response surface sets are indexed from NSOL+1 to NSOL+NRSS. Note that some options listed below apply only to solution sets (i.e., where 1 $\leq$ *n* $\leq$ NSOL) and some apply only to response surface sets (i.e., where NSOL+1 $\leq$ *n* $\leq$ NSOL+NRSS).

***GET,*Par*, PDS, *n*, Item1, IT1NUM, Item2, IT2NUM**

Item1	IT1NUM	Description
RSEQ	*i*	*Item2* = TTYP; *IT2NUM=j*; Type of the *j*-th regression term of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). IT2NUM=*j* ranges from 1 to NTRM (see Item1=RSEQ, Item2=NTRM). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS). Possible return values are: 1 = term is a constant (this term does not involve any random input variables) 2 = term is a linear term (this term involves only one random input variable) 3 = term is a purely quadratic term involving only one random input variable (this term involves only one random input variable) 4 = term is a mixed quadratic term involving two random input variables (this term involves two random input variables)
RSEQ	*i*	*Item2* = XRV1; *IT2NUM=j*; Index of the first random input variable involved in the *j*-th regression term of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS). An error appears if the term does not involve a random input variable, i.e. if the term is a constant.
RSEQ	*i*	*Item2* = XRV2; *IT2NUM=j*; Index of the second random input variable involved in the *j*-th regression term of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS). An error appears if the term does not involve a second random input variable, i.e. if the term is not a mixed quadratic term.
RSEQ	*i*	*Item2* = COEF; *IT2NUM=j*; Regression coefficient of the *j*-th regression term of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSEQ	*i*	*Item2* = SLOP; *IT2NUM=j*; Scaling slope of the *j*-th random input variable of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).
RSEQ	*i*	*Item2* = ICPT; *IT2NUM=j*; Scaling intercept of the *j*-th random input variable of the response surface equation for the *i*-th fitted random output parameter in the *n*-th result set. IT1NUM=*i* ranges from 1 to NFRP (see Item1=RSST, Item2=NFRP). This applies only to response surface sets (i.e., *n* ranges from NSOL+1 to NSOL+NRSS).

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Prob Design>Prob Method>Response Surface
Main Menu>Solution>Define Loads>Delete>Structural>Section
Utility Menu>Parameters>Get Scalar Data

***GO**, *Base*

Causes a specified line on the input file to be read next.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

Base

"Go to" action:

:*label* --

A user-defined label (beginning with a colon (:), 8 characters maximum). The command reader will skip (and wrap to the beginning of the file, if necessary) to the first line that begins with the matching :*label*.

Caution: This label option may not be mixed with do-loop or if-then-else constructs.

STOP --

This action will cause an exit from the ANSYS program at this line.

Command Default

Read lines sequentially.

Notes

Causes the next read to be from a specified line on the input file. Lines may be skipped or reread. The ***GO** command will *not* be executed unless it is part of a macro, user file (processed by ***USE**), an alternate input file (processed by **/INPUT**), or unless it is used in a batch-mode input stream. Jumping into, out of, or within a do-loop or an if-then-else construct to a :*label* line is not allowed.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*IF, *VAL1*, *Oper1*, *VAL2*, *Base1*, *VAL3*, *Oper2*, *VAL4*, *Base2*

Conditionally causes commands to be read.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

VAL1

First numerical value (or parameter which evaluates to a numerical value) in the conditional comparison operation. *VAL1*, *VAL2*, *VAL3* and *VAL4* can also be character strings (enclosed in quotes) or parameters for *Oper* = EQ and NE only.

Oper1

Operation label. A tolerance of 1.0E-10 is used for comparisons between real numbers:

EQ --
Equal (for *VAL1* = *VAL2*).

NE --
Not equal (for *VAL1* ≠ *VAL2*).

LT --
Less than (for *VAL1* < *VAL2*).

GT --
Greater than (for *VAL1* > *VAL2*).

LE --
Less than or equal (for *VAL1* ≤ *VAL2*).

GE --
Greater than or equal (for *VAL1* ≥ *VAL2*).

ABLT --
Absolute values of *VAL1* and *VAL2* before < operation.

ABGT --
Absolute values of *VAL1* and *VAL2* before > operation.

VAL2

Second numerical value (or parameter which evaluates to a numerical value) in the conditional comparison operation.

Base1

Action based on the logical expression (*Oper1*) being true. If false, continue reading at the next line. This is conditional, excepting the IF-THEN-ELSE constructs described below; any of the following constructs (through *Base1* = THEN) cause all subsequent fields to be ignored:

:*label* --
A user-defined label (beginning with a colon (:), 8 characters maximum). The command reader will skip (and wrap to the beginning of the file, if necessary) to the first line that begins with the matching :*label*.

Caution: This label option may not be mixed with do-loop or if-then-else constructs.

STOP --
This action will cause an exit from the ANSYS program at this line, unless running in interactive mode. In interactive mode, the program will not stop.

EXIT --
Exit the current do-loop [***EXIT**].

CYCLE --
Skip to the end of the current do-loop [***CYCLE**].

THEN --
Make this ***IF** an if-then-else construct (see below).

The following optional values determine the connection between the two logical clauses *Oper1* and *Oper2*

AND --
True if both clauses (*Oper1* and *Oper2*) are true.

OR --
True if either clause is true.

XOR --
True if either (but not both) clause is true.

VAL3
Third numerical value (or parameter which evaluates to a numerical value).

Oper2
Operation label. This will have the same labels as *Oper1*, except it uses *Val3* and *Val4*.

VAL4
Fourth Numerical value (or parameter value which evaluates to a numerical value).

Base2
Action based on the logical expression (*Oper1* and *Oper2*) being true. They will be the same values as *Base1*, except as noted.

Command Default

Read commands sequentially.

Notes

Conditionally causes commands to be read from a specific block or at a specific location. Twenty levels of nested ***IF** blocks are allowed. Jumping to a :*label* line is not allowed with keyboard entry. Jumping into, out of, or within a do-loop or an if-then-else construct to a :*label* line is not allowed.

The following is an example of an if-then-else construct:

***IF**,*VAL1*,*Oper*,*VAL2*,THEN

***ELSEIF**,*VAL1*,*Oper*,*VAL2*

***ELSEIF**,*VAL1*,*Oper*,*VAL2*

***ELSE**

***ENDIF**

where "----" represents a block of any number of commands. Any number of ***ELSEIF** clauses (or none) may be included (in the location shown). One ***ELSE** clause (at most) may be included (in the location shown). The ***IF** command is executed by evaluating its logical expression. If it is true, the block of commands following it is executed. The construct is considered to be complete and the command following the ***ENDIF** is executed next. If the logical expression is false, the next ***ELSEIF** command (if any) following the block is executed. The execution logic is the same as for ***IF**. The effect is that the logical expressions in the ***IF** and the ***ELSEIF** commands are sequentially tested until one is found to be true. Then the block of commands immediately following the expression is executed, which completes the execution of the if-then-else construct. If all ***IF** and ***ELSEIF** expressions are false, the block following the ***ELSE** command is executed, if there is one. Only one block of commands (at most) is executed within the if-then-else construct. If a batch input stream hits an end-of-file during a false ***IF** condition, the ANSYS run will not terminate normally. You will need to terminate it externally (use either the UNIX "kill" function or the Windows task manager). The ***IF**, ***ELSEIF** , ***ELSE**, and ***ENDIF** commands for each if-then-else construct must all be read from the same file (or keyboard).

This command is valid in any processor.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Solution>Define Loads>Delete>Structural>Section

/INQUIRE, *StrArray*, *FUNC*

Returns system information to a parameter.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

StrArray

Name of the "string array" parameter that will hold the returned values. String array parameters are similar to character arrays, but each array element can be as long as 128 characters. If the string parameter does not exist, it will be created.

FUNC

Specifies the type of system information returned:

LOGIN --
Returns the pathname of the login directory on UNIX systems or the pathname of the default directory (including drive letter) on Windows systems.

DOCU --
Returns the pathname of the ANSYS docu directory.

APDL --
Returns the pathname of the ANSYS APDL directory.

PROG --
Returns the pathname of the ANSYS executable directory.

AUTH --
Returns the pathname of the directory in which the license file resides.

USER --
Returns the name of the user currently logged-in.

DIRECTORY --
Returns the pathname of the current directory.

JOBNAME --
Returns the current *Jobname*. The value of *Jobname* can be up to 250 characters in length.

Returning the Value of an Environment Variable to a Parameter

If *FUNC*=ENV, the command format is **/INQUIRE**,*StrArray*,ENV,*ENVNAME*,*Substring*. In this instance, ENV specifies that the command should return the value of an environment variable. The following defines the remaining fields:

ENVNAME
Specifies the name of the environment variable.

Substring
If *Substring* = 1, the first substring (up to the first colon (:)) is returned. If *Substring* = 2, the second substring is returned, etc. For Windows platforms, the separating character is semicolon (;). If this argument is either blank or 0, the entire value of the environment variable is returned.

Returning the Value of a Title to a Parameter

If *FUNC* = TITLE, the command format is **/INQUIRE**,*StrArray*,TITLE,*Title_num*. In this context, the value of *Title_num* can be blank or 1 through 5. If the value is 1 or blank, the title is returned. If the value is 2 through 5, a corresponding subtitle is returned (2 denoting the first subtitle, and so on).

Returning Information About a File to a Parameter

The **/INQUIRE** command can also return information about specified files within the file system. For these capabilities, the format is **/INQUIRE**,*Parameter*,*FUNC*,*Fname*, *Ext*, *--*. The following defines the fields:

Parameter
Name of the parameter that will hold the returned values.

FUNC
Specifies the type of file information returned:

EXIST --
Returns a 1 if the specified file exists, and 0 if it does not.

DATE --
Returns the date stamp of the specified file in the format *yyyymmdd.hhmmss*.

SIZE --
Returns the size of the specified file in MB.

WRITE --
Returns the status of the write attribute. A 0 denotes no write permission while a 1 denotes write permission.

READ --
Returns the status of the read attribute. A 0 denotes no read permission while a 1 denotes read permission.

EXEC --
Returns the status of the execute attribute (this has meaning only in UNIX). A 0 denotes no execute permission while a 1 denotes execute permission.

LINES --
Returns the number of lines in an ASCII file.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Ext
Filename extension (8 character maximum).

--
Unused field

Notes

The **/INQUIRE** command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

/MAIL, --, *Address*, *Fname*, *Ext*

Mails file to the specifed address.

APDL: Macro Files

MP ME ST DY <> PR EM EH FL PP ED

--
Unused field.

Address
Email address (up to 64 characters) of the intended recipient of the file.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Ext
Filename extension (8 character maximum).

Notes

Issue the **/MAIL** command to alert someone when a long-running job has completed, as shown in this example:

```
...
SOLVE
/MAIL,,yourname@yourdomain.com,jobdone,txt
```

Menu Paths

This command cannot be accessed from a menu.

***MFOURI**, *Oper*, *COEFF*, *MODE*, *ISYM*, *THETA*, *CURVE*

Calculates the coefficients for, or evaluates, a Fourier series.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

Oper

Type of Fourier operation:

FIT --

Calculate Fourier coefficients *COEFF* from *MODE*, *ISYM*, *THETA*, and *CURVE*.

EVAL --

Evaluate the Fourier curve *CURVE* from *COEFF*, *MODE*, *ISYM* and *THETA*

COEFF

Name of the array parameter vector containing the Fourier coefficients (calculated if *Oper* = FIT, required as input if *Oper* = EVAL). See ***SET** for name restrictions.

MODE

Name of the array parameter vector containing the mode numbers of the desired Fourier terms.

ISYM

Name of the array parameter vector containing the symmetry key for the corresponding Fourier terms. The vector should contain keys for each term as follows:

0 or 1 --

Symmetric (cosine) term

-1 --

Antisymmetric (sine) term.

THETA, *CURVE*

Names of the array parameter vectors containing the theta vs. curve description, respectively. Theta values should be input in degrees. If *Oper* = FIT, one curve value should be supplied with each theta value. If *Oper* = EVAL, one curve value will be calculated for each theta value.

Notes

Calculates the coefficients of a Fourier series for a given curve, or evaluates the Fourier curve from the given (or previously calculated) coefficients. The lengths of the *COEFF*, *MODE*, and *ISYM* vectors must be the same--typically two times the number of modes desired, since two terms (sine and cosine) are generally required for each mode. The lengths of the *CURVE* and *THETA* vectors should be the same or the smaller of the two will be used. There should be a sufficient number of points to adequately define the curve--at least two times the number of coefficients. A starting array element number (1) must be defined for each array parameter vector. The vector specifications ***VLEN**, ***VCOL**, ***VABS**, ***VFACT**, and ***VCUM** do not apply to this command. Array elements should not be skipped with the ***VMASK** and the *NINC* value of the ***VLEN** specifications. The vector being calculated (*COEFF* if *Oper* is FIT, or *CURVE* if *Oper* is EVAL) must exist as a dimensioned array [***DIM**].

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Matrix Fourier

---

*MFUN, *ParR*, *Func*, *Par1*

Copies or transposes an array parameter matrix.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the resulting array parameter matrix. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Func
Copy or transpose function:

COPY --
Par1 is copied to *ParR*

TRAN --
Par1 is transposed to *ParR*. Rows (m) and columns (n) of *Par1* matrix are transposed to resulting *ParR* matrix of shape (n,m).

Par1
Array parameter matrix input to the operation.

Notes

Operates on one input array parameter matrix and produces one output array parameter matrix according to:

ParR = f(*Par1*)

where the function (f) is either a copy or transpose, as described above.

Functions are based on the standard FORTRAN definitions where possible. *ParR* may be the same as *Par1*. Starting array element numbers must be defined for each array parameter matrix. For example, ***MFUN**,A(1,5),COPY,B(2,3) copies matrix B (starting at element (2,3)) to matrix A (starting at element (1,5)). The diagonal corner elements for each submatrix must be defined: the upper left corner by the array starting element (on this command), the lower right corner by the current values from the ***VCOL** and ***VLEN** commands. The default values are the (1,1) element and the last element in the matrix. No operations progress across matrix planes (in the 3rd dimension). Absolute values and scale factors may be applied to all parameters [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. Array elements should not be skipped with the ***VMASK** and the *NINC* value of the ***VLEN** specifications. The number of rows [***VLEN**] applies to the *Par1* array. See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Matrix Functions

***MOPER**, *ParR*, *Par1*, *Oper*, *Par2*, *Par3*, *kDim*, --, *kOut*

Performs matrix operations on array parameter matrices.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR

The name of the resulting array parameter matrix. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Par1

First array parameter matrix input to the operation. For *Oper* = MAP, this is an *N x 3* array of coordinate locations at which to interpolate. *ParR* will then be an *N(out) x M* array containing the interpolated values.

Oper

Matrix operations:

INVERT --

(***MOPER**,*ParR*,*Par1*,INVERT)

Square matrix invert: Inverts the *n* x *n* matrix in *Par1* into *ParR*. The matrix must be well conditioned.

Warning: Non-independent or ill-conditioned equations can cause erroneous results.

MULT --

(***MOPER**,*ParR*,*Par1*,MULT,*Par2*)

Matrix multiply: Multiplies *Par1* by *Par2*. The number of rows of *Par2* must equal the number of columns of *Par1* for the operation.

COVAR --

(***MOPER**,*ParR*,*Par1*,COVAR,*Par2*)

Covariance: The measure of association between two columns of the input matrix (*Par1*). *Par1*, of size m runs (rows) by n data (columns) is first processed to produce a row vector containing the mean of each column which is transposed to a column vector (*Par2*) of n array elements. The *Par1* and *Par2* operation then produces a resulting n x n matrix (*ParR*) of covariances (with the variances as the diagonal terms).

CORR --

(***MOPER**,*ParR*,*Par1*,CORR,*Par2*)

Correlation: The correlation coefficient between two variables. The input matrix (*Par1*), of size m runs (rows) by n data (columns), is first processed to produce a row vector containing the mean of each column which is then transposed to a column vector (*Par2*) of n array elements. The *Par1* and *Par2* operation then produces a resulting *n x n* matrix (*ParR*) of correlation coefficients (with a value of 1.0 for the diagonal terms).

SOLV --

(***MOPER**,*ParR*,*Par1*,SOLV,*Par2*)

Solution of simultaneous equations: Solves the set of *n* equations of *n* terms of the form $a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$ where *Par1* contains the matrix of a-coefficients, *Par2* the vector(s) of b-values, and *ParR* the vector(s) of x-results. *Par1* must be a square matrix. The equations must be linear, independent, and well conditioned.

Warning: Non-independent or ill-conditioned equations can cause erroneous results.

SORT --

(***MOPER**,*ParR*,*Par1*,SORT,*Par2*)

Matrix sort: Sorts matrix *Par1* according to sort vector*Par2* and places the result in *Par1*. Rows of *Par1* are moved to the corresponding positions indicated by the values of *Par2*. *Par2* may be a column of *Par1* (in which case it will also be reordered). *ParR* is the vector of initial row positions. Sorting *Par1* according to *ParR* should reproduce the initial ordering.

NNEAR --

(***MOPER**,*ParR*,*Par1*,NNEAR,Toler)

Nearest Node: Quickly determine all the nodes within a specified tolerance of a given array.

ParR is a vector of the nearest selected nodes, or 0 if no nodes are nearer than Toler. *Par1* is the *n x 3* array of coordinate locations.

ENEAR --

(***MOPER**,*ParR*,*Par1*,ENEAR,Toler)

Nearest Element: Quickly determine the elements whose centroids are within a specified tolerance of the points in a given array.

ParR is a vector of the nearest selected elements, or 0 if no element centroids are nearer than Toler. *Par1* is the *n x 3* array of coordinate locations.

MAP --

(***MOPER**,*ParR*,*Par1*,MAP,*Par2*,*Par3*,*kDim*,,*kOut*)

Maps the results from another program onto your ANSYS finite element model. For example, you can map pressures from a CFD analysis onto your model for a structural analysis.

When you map results, the subsequent *Par2* and *Par3* arguments define your input values and their locations, and the arguments that follow determine the search area and interpolation schemes (see below).

For *0per* = MAP, output points are incorrect if they are not within the boundaries (area or volume) set via the specified input points. Also, calculations for out-of-bound points require much more processing time than do points that are within bounds.

When mapping results from one analysis to another (*0per* = MAP), *Par1* will be your final, *N(out) x 3* array of points. *Par2*will be an *N(in) x M* array that corresponds to the points in*Par3*. For each point in the destination mesh, all possible triangles in the source mesh are searched to find the best triangle containing each point. It then does a linear interpolation inside this triangle. You should carefully specify your interpolation method and search criteria in order to provide faster and more accurate results. Results mapping (*0per* = MAP) is available from the command line only.

Par2
Second array parameter matrix input to the operation. For the COVAR and CORR operations, this parameter must exist as a dimensioned array vector without specified values since its values (means) will be calculated as part of the operations. For MAP, this will be an {*N(in) x M*} array of values to be interpolated, where N(in) is the number of points to interpolate from, and M is the number of values at each point. For the ENEAR and NNEAR operations, this parameter specifies the tolerance for the search.

Par3
Third array parameter, used for *Oper* = MAP. This is an *N x 3* array of coordinate locations corresponding to the values in *Par2*.

kDim
Interpolation criteria; used for *Oper* = MAP:

If *kDim* = 2 or 0, two dimensional interpolation is applied (interpolate on a surface).
If *kDim* = 3, three dimensional interpolation is applied (interpolate on a volume).

--
Unused field

kOut
Outside region results; used for *Oper* = MAP

If *kOut* = 0, use the value(s) of the nearest region point for points outside of the region.
If *kOut* = 1, set results extrapolated outside of the region to zero.

Notes

Each array starting element number must be defined for each array parameter matrix. For example, ***MOPER**,A(2,3),B(1,4),MULT,C(1,5) multiplies submatrix B (starting at element (1,4)) by submatrix C (starting at element (1,5)) and puts the result in matrix A (starting at element (2,3)).

The diagonal corner elements for each submatrix must be defined: the upper left corner by the array starting element (on this command), the lower right corner by the current values from the ***VCOL** and ***VLEN** commands. The default values are the (1,1) element and the last element in the matrix. No operations progress across matrix planes (in the 3rd dimension). Absolute values and scale factors may be applied to all parameters [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. Array elements should not be skipped with the ***VMASK** and the *NINC* value of the ***VLEN** specifications. See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Matrix Operations

***MSG**, *Lab*, *VAL1*, *VAL2*, *VAL3*, *VAL4*, *VAL5*, *VAL6*, *VAL7*, *VAL8*

Writes an output message via the ANSYS message subroutine.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Lab
Label for output and termination control:

INFO --
Writes the message with no heading (default).

NOTE --
Writes the message with a "NOTE" heading.

WARN --
Writes the message with a "WARNING" heading. Also writes the message to the errors file, **Jobname.ERR**.

ERROR --
Writes the message with a "ERROR" heading and causes run termination (if batch) at earliest "clean exit" point. Also writes the message to the errors file, **Jobname.ERR**.

FATAL --
Writes the message with a "FATAL ERROR" heading and causes run termination immediately. Also writes the message to the errors file, **Jobname.ERR**.

UI --
Writes the message with a "NOTE" heading and displays it in the message dialog box. This option is most useful in GUI mode.

VAL1, *VAL2*, *VAL3*, *VAL4*, *VAL5*, *VAL6*, *VAL7*, *VAL8*
Numeric or alphanumeric character values to be included in message. Values may be the results of parameter evaluations. All numeric values are assumed to be double precision. The FORTRAN nearest integer (NINT) function is used to form integers for the %I specifier.

Notes

Allows writing an output message via the ANSYS message subroutine. Also allows run termination control. This command is used only when contained in a prepared file read into the ANSYS program (i.e., ***USE**,**/INPUT**, etc.). A message format must immediately follow the ***MSG** command (on a separate line, without parentheses, as described below).

The message format may be up to 80 characters long, consisting of text strings and predefined "data descriptors" between the strings where numeric or alphanumeric character data are to be inserted. The normal descriptors are %I for integer data, %G for double precision data, %C for alphanumeric character data, and %/ for a line break. The corresponding FORTRAN data descriptors are I9, 1PG16.9 and A8, respectively. Each descriptor must be preceded by a blank. There must be one data descriptor for each specified value (8 maximum) in the order of the specified values.

Enhanced descriptions may also be used:

%w.pE	w is field width
%w.pG	p is precision
%w.pF	
%%	a single percent sign
%wC; %wS	character string
%-wC; %-wS	left justify string
%wX	w blank characters
%wI	integer format
%0wI	pad integer with leading zeros rather than blanks
%0w.pI	w is field width; p is number of characters filled

Do not begin ***MSG** format lines with ***IF**, ***ELSE** , ***ELSEIF** , or ***ENDIF** . If the last nonblank character of the message format is an ampersand (&), a second line will also be read as a continuation of the format. Up to nine continuations (ten total lines) may be read. If normal descriptions are used, then consecutive blanks are condensed into one blank upon output, and a period is appended. Up to ten lines of output of 72 characters each may be produced (using the %/ descriptor). Two examples follow.

Here is an example of the ***MSG** command and a format to print a message with two integer values and one real value:

```
*MSG, INFO, 'Inner',25,1.2,148
Radius ( %C) = %I, Thick = %G, Length = %I
```

The output line is:

```
Radius (Inner) = 25, Thick = 1.2, Length = 148.
```

Here is an example illustrating multiline displays in GUI message windows:

```
*MSG,UI,Vcoilrms,THTAv,Icoilrms,THTAi,Papprnt,Pelec,PF,indctnc
Coil RMS voltage, RMS current, apparent pwr, actual pwr, pwr factor: %/&
Vcoil = %G V (electrical angle = %G DEG) %/&
Icoil = %G A (electrical angle = %G DEG) %/&
APPARENT POWER = %G W %/&
ACTUAL POWER = %G W %/&
Power factor: %G %/&
Inductance = %G %/&
VALUES ARE FOR ENTIRE COIL (NOT JUST THE MODELED SECTOR)
```

Note — The **/UIS**,MSGPOP command controls which messages are displayed in the message dialog box when the GUI is active. All messages produced by the ***MSG** command are subject to the **/UIS** specification, with one exception, If *Lab* = UI, the message will be displayed in the dialog box regardless of the **/UIS** specification.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

***MWRITE**, *ParR*, *Fname*, *Ext*, --, *Label*, *n1*, *n2*, *n3*

Writes a matrix to a file in a formatted sequence.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR

The name of the array parameter. See ***SET** for name restrictions.

Fname

File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

If the file name fields are left blank, the default file is the current output file.

Ext

Filename extension (8 character maximum).

- -
Unused field

Label
Can use a value of IJK, IKJ, JIK, JKI, KIJ, KJI, or blank (JIK).

n1, *n2*, *n3*
Write as (((*ParR*(i,j,k), k = 1,n1), i = 1, n2), j = 1, n3) for *Label* = KIJ. *n1, n2,* and *n3* default to the corresponding dimensions of the array parameter ParR.

Notes

Writes a matrix or vector to a specified file in a formatted sequence. You can also use the ***VWRITE** command to write data to a specified file. Both commands contain format descriptors on the line immediately following the command. The format descriptors can be in either Fortran or C format.

Fortran format descriptors are enclosed in parentheses. They must immediately follow the ***MWRITE** command on a separate line of the same input file. The word FORMAT should not be included. The format must specify the number of fields to be written per line, the field width, the placement of the decimal point, etc. There should be one field descriptor for each data item written. The write operation uses the available system FORTRAN FORMAT conventions (see your system FORTRAN manual). Any standard FORTRAN *real* format (such as (4F6.0), (E10.3,2X,D8.2), etc.) and character format (A) may be used. Integer (I) and list-directed (*) descriptors may *not* be used. Text may be included in the format as a quoted string. *The FORTRAN descriptor must be enclosed in parentheses* and the format must not exceed 80 characters (including parentheses).

The "C" format descriptors are used if the first character of the format descriptor line is not a left parenthesis. "C" format descriptors may be up to 80 characters long, consisting of text strings and predefined "data descriptors" between the strings where numeric or alphanumeric character data are to be inserted. The normal descriptors are %I for integer data, %G for double precision data, %C for alphanumeric character data, and %/ for a line break. Each descriptor must be preceded by a blank. There must be one data descriptor for each specified value in the order of the specified values. The enhanced formats described in ***MSG** may also be used.

The starting array element number must be defined. Looping continues in the directions indicated by the Label argument. The number of loops and loop skipping may also be controlled with the ***VLEN** and ***VMASK** commands. These commands work in the N1 direction. The vector specifications ***VABS**, ***VFACT**, and ***VCUM** do not apply to this command. See the ***VOPER** command for details. If you are in the GUI the ***MWRITE** command must be contained in an externally prepared file and read into ANSYS (i.e., ***USE**, **/INPUT**, etc.).

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Parameters>Write to File

PARRES, *Lab*, *Fname*, *Ext*, --
Reads parameters from a file.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Lab
Read operation:

NEW --
Replace current parameter set with these parameters (default).

CHANGE --
Extend current parameter set with these parameters, replacing any that already exist.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

The file name defaults to **Jobname**.

Ext
Filename extension (8 character maximum).

The extension defaults to PARM if *Fname* is blank.

--
Unused field

Notes

Reads parameters from a coded file. The parameter file may have been written with the **PARSAV** command. The parameters read may replace or change the current parameter set.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Restore Parameters

PARSAV, *Lab*, *Fname*, *Ext*, --

Writes parameters to a file.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Lab
Write operation:

SCALAR --
Write only scalar parameters (default).

ALL --
Write scalar and array parameters. Parameters may be numeric or alphanumeric.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

The file name defaults to **Jobname**.

Ext
Filename extension (8 character maximum).

The extension defaults to PARM if *Fname* is blank.

- -

Unused field

Notes

Writes the current parameters to a coded file. Previous parameters on this file, if any, will be overwritten. The parameter file may be read with the **PARRES** command.

PARSAV/**PARRES** operations truncate some long decimal strings, and can cause differing values in your solution data when other operations are performed. A good practice is to limit the number of decimal places you will use before and after these operations.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Save Parameters

/PMACRO

Specifies that macro contents be written to the session log file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Notes

This command forces the contents of a macro or other input file to be written to **Jobname.LOG**. It is valid only within a macro or input file, and should be placed at the top of the file. **/PMACRO**should be included in any macro or input file that calls GUI functions.

Menu Paths

This command cannot be accessed from a menu.

/PSEARCH, *Pname*

Specifies a directory to be searched for "unknown command" macro files.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Pname

Path name (64 characters maximum, and must include the final delimiter) of the middle directory to be searched. Defaults to the user home directory. If *Pname* = OFF, search only the ANSYS and current working directories. If *Pname* = STAT, list the current middle directory and show the **ANSYS_MACROLIB** setting.

Command Default

The middle directory searched is the user home directory.

Notes

Specifies the pathname of a directory for file searches when reading "unknown command" macro files. The search for the files is typically from the ANSYS directory, then from the user home directory, and then from the current working directory. This command allows the middle directory searched to be other than the user home directory.

This command is valid only at the Begin Level.

Menu Paths

Utility Menu>Macro>Macro Search Path

***REPEAT**, *NTOT*, *VINC1*, *VINC2*, *VINC3*, *VINC4*, *VINC5*, *VINC6*, *VINC7*, *VINC8*, *VINC9*, *VINC10*, *VINC11*
Repeats the previous command.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

NTOT
Number of times the preceding command is executed (including the initial execution). Must be 2 or greater. *NTOT* of 2 causes one repeat (for a total of 2 executions).

VINC1, *VINC2*, *VINC3*, *VINC4*, *VINC5*, *VINC6*, *VINC7*, *VINC8*, *VINC9*, *VINC10*, *VINC11*
Value increments applied to first through eleventh data fields of the preceding command.

Notes

***REPEAT** must immediately follow the command that is to be repeated. The numeric arguments of the initial command may be incremented in the generated commands. The numeric increment values may be integer or real, positive or negative, zero or blank. Alphanumeric arguments cannot be incremented. For large values of *NTOT*, consider printout suppression (**/NOPR** command) first.

Most commands beginning with slash (/), star (*), as well as "unknown command" macros, cannot be repeated. For these commands, or if more than one command is to be repeated, include them within a do-loop. Graphics slash commands are an exception and can be repeated. Commands causing file switching (causing additional commands to be read) cannot be repeated. If a ***REPEAT** command immediately follows another ***REPEAT** command, the repeat action only applies to the last non-***REPEAT** command. Also, ***REPEAT** should not be used in interactive mode immediately after a) a command (or its log file equivalent) that uses picking, or b) a command that requires a response from the user.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*RETURN, *Level*

Returns input stream to a higher level.

APDL: Process Controls

MP ME ST DY <> PR EM EH FL PP ED

Level

Number of levels to move up from the current level.

Negative --

Move relative to current level. For example: *Return,-2 will go up two levels from the current level.

Positive --

Move to absolute level. For example: *Return,2 will go to level 2.

Level 0 is the primary input file.

Notes

This command is used to jump to the macro call sequence, ending the current macro file, and returning to the line after the calling line in the previous file. Unlike the ***GO** command, this command may be used inside ***IF** or ***DO** constructs.

Menu Paths

This command cannot be accessed from a menu.

---

*SET, *Par*, *VALUE*, *VAL2*, *VAL3*, *VAL4*, *VAL5*, *VAL6*, *VAL7*, *VAL8*, *VAL9*, *VAL10*

Assigns values to user-named parameters.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par

An alphanumeric name used to identify this parameter. *Par* may be up to 32 characters, beginning with a letter and containing only letters, numbers, and underscores. Examples: ABC A3X TOP_END. ANSYS command names, function names, label names, component and assembly names, etc., should not be used. Parameter names beginning with an underscore (e.g., _LOOP) are reserved for use by ANSYS and should be avoided. Parameter names ending in an underscore are not listed by the ***STATUS** command. Array parameter names must be followed by a subscript, and the entire expression must be 32 characters or less. Examples: A(1,1) NEW_VAL(3,2,5) RESULT(1000). There is no character parameter substitution for the *Par* field. Table parameters that are used in command fields (where constant values are normally given) are limited to 32 characters.

VALUE

Numerical value or alphanumeric character string (up to 8 characters enclosed in single quotes) to be assigned to this parameter. Examples: A(1,3)=7.4 B='ABC3'. May also be a parameter or a parametric expression. Examples: C=A(1,3) A(2,2)=(C+4)/2. If blank, delete this parameter. Example: A= deletes parameter A.

VAL2, *VAL3*, *VAL4*, *VAL5*, *VAL6*, *VAL7*, *VAL8*, *VAL9*, *VAL10*

If *Par* is an array parameter, values *VAL2* through *VAL10* (up to the last nonblank value) are sequentially assigned to the succeeding array elements of the column. Example: ***SET**,A(1,4),10,11 assigns A(1,4)=10, A(2,4)=11. ***SET**,B(2,3),'file10','file11' assigns B(2,3)='file10', B(3,3)='file11'.

Notes

Assigns values to user-named parameters that may be substituted later in the run. The equivalent (and recommended) format is

Par = *VALUE*,*VAL2*,*VAL3*,*VAL4*,*VAL5*,*VAL6*,*VAL7*,*VAL8*,*VAL9*,*VAL10*

which may be used in place of ***SET**,*Par*, ... for convenience.

This command is valid in any processor.

Parameter Definitions

Parameters (numeric or character) may be scalars (single valued) or arrays (multiple valued in one, two, or three dimensions). Up to 5000 unique parameter names may be defined in any ANSYS run (fewer than 5000 are available due to GUI and ANSYS macro requirements); however, a single array parameter name can represent any number of values. Parameter values may be redefined at any time. Array parameters may also be assigned values within a do-loop [***DO**] for convenience. Internally programmed do-loop commands are also available with the ***V*XX** commands (***VFILL**). Parameter values (except for parameters ending in an underscore) may be listed with the ***STATUS** command, displayed with the ***VPLOT** command (numeric parameters only), and modified with the ***VEDIT** command (numeric parameters only). Parameters can also be resolved in comments created by the **/COM** command (see **/COM** for complete documentation). A parameter can be deleted by redefining it with a blank *VALUE*. If the parameter is an array, the entire array is deleted. Parameters may also be defined by a response to a query with the ***ASK** command or from an "ANSYS-supplied" value with the ***GET** command.

Array Parameters

Array parameters must be dimensioned [***DIM**] before being assigned values. Scalar parameters that are not defined are initialized to a "near" zero value. Numeric array parameters are initialized to zero when dimensioned, and character array parameters are initialized to blank. An existing array parameter must be deleted before it can be redimensioned. Array parameter names must be followed by a subscript list (enclosed in parentheses) identifying the element of the array. The subscript list may have one, two, or three values (separated by commas). Typical array parameter elements are A(1,1), NEW_VAL(3,2,5), RESULT(1000). Subscripts for defining an array element must be integers (or parameter expressions that evaluate to integers). Non-integer values are rounded to the nearest integer value. All array parameters are stored as 3-D arrays with the unspecified dimensions set to 1. For example, the 4th array element of a 1-dimensional array, A(4), is stored as array element A(4,1,1). Arrays are patterned after standard FORTRAN conventions.

Numerical Parameter Substitution

If the parameter name *Par* is input in a numeric argument of a command, the numeric value of the parameter (as assigned with ***SET**, ***GET**, =, etc.) is substituted into the command at that point. Substitution occurs only if the parameter name is used between blanks, commas, parentheses, or arithmetic operators (or any combination) in a numeric argument. Substitution can be prevented by enclosing the parameter name *Par* within single quotes ('), if the parameter is alone in the argument; if the parameter is part of an arithmetic expression, the entire expression must be enclosed within single quotes to prevent substitution. In either case the character string will be used instead of the numeric value (and the string will be taken as 0.0 if it is in a numeric argument).

A forced substitution is available in the text fields of the **/TITLE**, **/STITLE**, **/TLABEL**, **/AN3D**, **/SYP** (*ARG1--ARG8*), and ***ABBR** commands by enclosing the parameter within percent (%) signs. Also, parameter substitution may be forced within the file name or extension fields of commands having these fields by enclosing the parameter

within percent (%) signs. Array parameters [***DIM**] must include a subscript (within parentheses) to identify the array element whose value is to be substituted, such as A(1,3). Out-of-range subscripts result in an error message. Non-integer subscripts are allowed when identifying a TABLE array element for substitution. A proportional linear interpolation of values among the nearest array elements is performed before substitution. Interpolation is done in all three dimensions.

Note — Interpolation is based upon the assigned index numbers which must be defined when the table is filled [***DIM**].

Character Parameter Substitution

Most alphanumeric arguments permit the use of character parameter substitution. When the parameter name *Par* input, the alphanumeric value of the parameter is substituted into the command at that point. Substitution can be suppressed by enclosing the parameter name within single quotes ('). Forced substitution is available in some fields by enclosing the parameter name within percent (%) signs. Valid forced substitution fields include command name fields, *Fname* (filename) or *Ext* (extension) arguments, ***ABBR** command (*Abbr* arguments), **/TITLE** and **/STITLE** commands (*Title* argument) and **/TLABEL** command (*Text* argument). Character parameter substitution is also available in the ***ASK**, **/AN3D**, ***CFWRITE**, ***IF**, ***ELSEIF**, ***MSG**, ***SET**, ***USE**, ***VREAD**, and ***VWRITE** commands. Character array parameters must include a subscript (within parentheses) to identify the array element whose value is to be substituted.

Parameter Expressions

If a parameter *operation expression* is input in a numeric argument, the numeric value of the expression is substituted into the command at that point. Allowable operation expressions are of the form

E1oE2oE3 ...oE10

where E1, E2, etc. are expressions connected by operators (o). The allowable operations (o) are

+ -- * / ** < >

For example, A+B**C/D*E is a valid operation expression. The * represents multiplication and the ** represents exponentiation.

Note — Exponentiation of a negative number (without parentheses) to an integer power follows standard FORTRAN hierarchy conventions; that is, the positive number is exponentiated and then the sign is attached. Thus, -4**2 is evaluated as -16. If parentheses are applied, such as (-4)**2, the result is 16.

A parameter is evaluated as a number within parentheses before exponentiation. Exponentiation of a negative number to a non-integer power is performed by exponentiating the positive number and prepending the minus sign, for example, -4**2.3 is -(4**2.3). The < and > operators allow conditional substitution. For example, E1<E2 substitutes the value of E1 if the comparison is true or the value of E2 if the comparison is false.

Spaces should not be used around operation symbols since " *" (a space and a star) makes the rest of the line a comment. Operation symbols (or symbols and signs) may not be immediately adjacent to each other. Parentheses may be used to separate symbols and signs, to determine a hierarchy of operations, or for clarity. For example, A**(-B) must be used instead of A**-B. Numbers ending with +0nn or -0nn are assumed to be of exponential form (as written on files by some computer systems) so that 123-002 is 123E-2 while 123-2 is 121. This form of exponential data should not be input directly. The default hierarchy follows the standard FORTRAN conventions, namely:

- operations in parentheses (innermost first)
- then exponentiation (right to left)
- then multiplication or division (left to right)
- then unary association (such as +A or -A)
- then addition or subtraction (left to right)
- then logical evaluations (left to right).

Expressions (E) may be a constant, a parameter, a function, or another operation expression (of the form E1oE2oE3 ...oE10). Functions are of the form FTN(A) where the argument (A) may itself be of the form E1oE2oE3 ...oE10. Operations are recursive to a level of four deep (three levels of internally nested parentheses). Iterative floating point parameter arithmetic should not be used for high precision input because of the accumulated numerical round off-error. Up to 10 expressions are accepted within a set of parenthesis.

Valid functions (which are based on standard FORTRAN functions where possible) are:

SIN(*X*)	Sine
COS(*X*)	Cosine
TAN(*X*)	Tangent
ASIN(*X*)	Arcsine
ACOS(*X*)	Arccosine
ATAN(*X*)	Arctangent
ATAN2(*Y,X*)	Arctangent (*Y*/*X*) with the sign of each component considered
SINH(*X*)	Hyperbolic sine
COSH(*X*)	Hyperbolic cosine
TANH(*X*)	Hyperbolic tangent
SQRT(*X*)	Square root
ABS(*X*)	Absolute value
SIGN(*X,Y*)	Absolute value of *X* with sign of *Y*. *Y*=0 results in positive sign
NINT(*X*)	Nearest integer
MOD(*X,Y*)	Remainder of *X*/*Y* *Y*=0 returns zero (0)
EXP(*X*)	Exponential
LOG(*X*)	Natural log
LOG10(*X*)	Common log
RAND(*X,Y*)	Random number, where *X* is the lower bound, and *Y* is the upper bound
GDIS(*X,Y*)	Random sample of Gaussian distributions, where *X* is the mean, and *Y* is the standard deviation
LWCASE(*CPARM*)	Lowercase equivalent of character parameter *CPARM*
UPCASE(*CPARM*)	Uppercase equivalent of character parameter *CPARM*
VALCHR(*CPARM*)	Numeric value of character parameter *CPARM* (If *CPARM* is a numeric parameter, returns 0.0)
CHRVAL(*PARM*)	Character value of numerical parameter *PARM*. For ABS(*PARM*) < 10, character value format is F8.5; for 10 $\leq$ ABS(*PARM*) < 1000, format is F8.3; for 1,000 $\leq$ ABS(*PARM*)< 10,000,000, format is F8.0. For 10,000,000 $\leq$ PARM < 100,000,000, format is also F8.0. Otherwise result is 0.0 and is not a character value.
IBSET(b1,n2)	Set the n2 bit in value b1 (bits are numbered from 0 to 31)

IBCLEAR(b1,n2)	Clear the n2 bit in value b1
BTEST(b1,n2)	Test the n2 bit in value b1 (return true (1.0) if bit is set)
BITAND(b1,b2)	Bitwise AND of value b1 and b2
BITOR(b1,b2)	Bitwise OR of value b1 and b2
BITXOR(b1,b2)	Bitwise XOR of value b1 and b2
BITSET(b1,b2)	Set the b2 bits in b1
BITCLEAR(b1,b2)	Clear the b2 bits in b1

Function arguments (*X*,*Y*, etc.) must be enclosed within parentheses and may be numeric values, parameters, or expressions. Input arguments for angular functions must evaluate to radians by default. Output from angular functions are also in radians by default. See the ***AFUN** command to use degrees instead of radians for the angular functions. See the ***VFUN** command for applying these parameter functions to a sequence of array elements. Additional functions, called "get functions" are described with the ***GET** command.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Delete>Structural>Section
Main Menu>Preprocessor>LS-DYNA Options>Inertia Options>Define Inertia
Main Menu>Preprocessor>Modeling>Create>Circuit>Builder>ROM>ElecStruc
Main Menu>ROM Tool>Mode Selection>Edit
Main Menu>Solution>Define Loads>Delete>Structural>Section
Main Menu>Solution>Solve>Electromagnet>Static Analysis>Induct Matrix
Main Menu>Solution>Time Controls>Time Step Prediction
Utility Menu>Parameters>Scalar Parameters

***SREAD**, *StrArray*, *Fname*, *Ext*, *--*, *nChar*, *nSkip*, *nRead*

Reads a file into a string array parameter.

APDL: Array Parameters

MP ME ST DY <> PR EM EH FL PP ED

StrArray
Name of the “string array” parameter which will hold the read file. String array parameters are similar to character arrays, but each array element can be as long as 128 characters. If the string parameter does not exist, it will be created. The array will be created as: *DIM,StrArray,STRING,nChar,nRead

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Ext
Filename extension (8 character maximum).

--
Unused field

nChar
Number of characters per line to read (default is length of the longest line in the file).

nSkip
Number of lines to skip at the start of the file (default is 0).

nLines
Number of lines to read from the file (default is the entire file).

Notes

The ***SREAD** command reads from a file into a string array parameter. The file must be an ASCII text file.

Menu Paths

This command cannot be accessed from a menu.

---

*STATUS, *Par*, *IMIN*, *IMAX*, *JMIN*, *JMAX*, *KMIN*, *KMAX*, *LMIN*, *LMAX*, *MMIN*, *MMAX*, *KPRI*

Lists the current parameters and abbreviations.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par
Specifies the parameter or sets of parameters listed. For array parameters, use *IMIN*, *IMAX*, etc. to specify ranges. Use ***DIM** to define array parameters. Use ***VEDIT** to review array parameters interactively. Use ***VWRITE** to print array values in a formatted output. If *Par* is blank, list all scalar parameter values, array parameter dimensions, and abbreviations. If *ARGX*, list the active set of local macro parameters (ARG1 to AR99) [***USE**].

The following are possible values for *Par*

ALL or blank --
Lists all parameters (except those with names beginning or ending with an underbar) and toolbar abbreviations.

_PRM --
Lists all parameters with names beginning with an underbar (_). These are ANSYS internal parameters.

PRM_ --
Lists all parameters with names ending with an underbar (_). A good APDL programming convention is to ensure that all parameters created by your system programmer are named with a trailing underbar.

ABBR --
Lists all toolbar abbreviations.

PARM --
Lists all user parameters.

PARNAME --
Lists only the parameter specified. *PARNAME* cannot be a parameter name beginning or ending with an underbar.

ARGX --
Lists all parameter values passed into the current macro (ARG1- AR18).

IMIN, *IMAX*, *JMIN*, *JMAX*, *KMIN*, *KMAX*, *LMIN*, *LMAX*, *MMIN*, *MMAX*
Range of array elements to display (in terms of the dimensions (row, column, plane, book, and shelf). Minimum values default to 1. Maximum values default to the maximum dimension values. Zero may be input for *IMIN*,

JMIN, and *KMIN* to display the index numbers. See ***TAXIS** command to list index numbers of 4- and 5-D tables.

KPRI
Use this field to list your primary variable labels (X, Y, Z, TIME, etc.).

1
List the labels (default). *YES*, *Y*, or *ON* are also valid entries.

0
Do not list the labels. *NO*, *N*, or *OFF* are also valid entries.

Notes

This command is valid in any processor.

Menu Paths

Utility Menu>List>Other>Named Parameter
Utility Menu>List>Other>Parameters
Utility Menu>List>Status>Parameters>All Parameters
Utility Menu>List>Status>Parameters>Named Parameters

***TAXIS**, *ParmLoc*, *nAxis*, *Val1*, *Val2*, *Val3*, *Val4*, *Val5*, *Val6*, *Val7*, *Val8*, *Val9*, *Val10*

Defines table index numbers.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParmLoc
Name and starting location in the table array parameter for indexing. Indexing occurs along the axis defined with *nAxis*.

nAxis
Axis along which indexing occurs. Valid labels are:

1 --
Corresponds to Row. Default.

2 --
Corresponds to Column.

3 --
Corresponds to Plane.

4 --
Corresponds to Book.

5 --
Corresponds to Shelf.

ALL --
Lists all index numbers. Valid only if *Val1* = LIST.

Val1 - Val10
Values of the index numbers for the axis *nAxis*, starting from the table array parameter location *ParmLoc*. You can define up to ten values.

To list the index values specified with *nAxis*, issue *Val1* = LIST. If *Val1* = LIST, *Val2* - *Val10* are ignored.

Notes

***TAXIS** is a convenient method to define table index values. These values reside in the zero column, row, etc. Instead of filling values in these zero location spots, use the ***TAXIS** command. For example,

```
*TAXIS,longtable(1,4,1,1),2,1.0,2.2,3.5,4.7,5.9
```

would fill index values 1.0, 2.2, 3.5, 4.7, and 5.9 in *nAxis* 2 (column location), starting at location 4.

To list index numbers, issue ***TAXIS**,*ParmLoc*, *nAxis*, LIST, where *nAxis* = 1 through 5 or ALL.

Menu Paths

This command cannot be accessed from a menu.

/TEE, *Label*, *Fname*, *Ext*, --

Writes a list of commands to a specified file at the same time that the commands are being executed.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Label
Indicates how ANSYS is to interpret this **/TEE** command:

NEW --
Signals the beginning of the command text that is to be written to *Fname*. If *Fname* already exists, specifying NEW causes the contents of *Fname* to be overwritten.

APPEND --
Indicates that you want to append to *Fname* the command text that follows.

END --
Signals the end of the command text that is to be written to or appended to *Fname*.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Ext
Filename extension (8 character maximum).

If you plan to execute the file as if it were an ANSYS command, use the extension *.mac*.

--
Unused field

Notes

You can use the **/TEE** command to record a macro to a specified file at the same time that the macro is being executed. It is similar to the UNIX **tee** command.

For more information about the **/TEE** command, see the *ANSYS APDL Programmer's Guide*.

The following example illustrates the use of the **/TEE** command. If you issue these commands:

```
/tee,new,myfile,mac
et,1,42,0,0,1
ex,1,3e7
/tee,end
/tee,append,myfile,mac
n,1,8
n,5,11
fill
ngen,5,5,1,5,1,0,1
/tee,end
```

the content of **myfile.mac** is:

```
et,1,42,0,0,1
ex,1,3e7
n,1,8
n,5,11
fill
ngen,5,5,1,5,1,0,1
```

This command is valid in any processor, but only during an interactive run.

Menu Paths

This command cannot be accessed from a menu.

---

*TOPER, *ParR*, *Par1*, *Oper*, *Par2*, *FACT1*, *FACT2*, *CON1*

Operates on table parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
Name of the resulting table parameter. The command will create a table array parameter with this name. Any existing parameter with this name will be overwritten.

Par1
Name of the first table parameter.

Oper
The operation to be performed: ADD. The operation is: ParR(i,j,k) = FACT1*Par1(i,j,k) + FACT2 *Par2(i,j,k) +CON1

Par2
Name of the second table parameter.

FACT1
The first table parameter multiplying constant. Defaults to 1.

FACT2
The second table parameter multiplying constant. Defaults to 1.

CON1
The constant increment for offset. Defaults to 0.

Notes

***TOPER** operates on table parameters according to: ParR(i,j,k) = FACT1*Par1(i,j,k) + FACT2 *Par2(i,j,k) +CON1

Par1 and Par2 must have the same dimensions and the same variable names corresponding to those dimensions. Par1 and Par2 must also have identical index values for rows, columns, etc.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Table Operations

***TREAD**, *Par*, *Fname*, *Ext*, *--*, *NSKIP*

Reads data from an external file into a table array parameter.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par
Table array parameter name as defined by the ***DIM** command.

Fname
File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

File name has no default.

Ext
Filename extension (8 character maximum).

Extension has no default.

--
Unused field

NSKIP
Number of comment lines at the beginning of the file being read that will be skipped during the reading. Default = 0.

Notes

Use this command to read in a table of data from an external file into an ANSYS table array parameter. The external file may be created using a text editor or by an external application or program. The external file must be in tab-delimited, blank-delimited, or comma-delimited format to be used by ***TREAD**. The ANSYS TABLE type array parameter must be defined before you can read in an external file. See ***DIM** for more information.

This command is not applicable to 4- or 5-D tables.

Menu Paths

Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>TimeInt>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>TimeInt>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>TimeInt>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>Voltage>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>Voltage>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>Voltage>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Boundary>Voltage>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>AppCharge>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>AppCharge>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>AppCurrent>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>AppCurrent>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>ImprCurr>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Electric>Excitation>ImprCurr>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Flow>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Flow>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Elements
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Heat Generat>Uniform Heat Gen
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Displacement>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Displacement>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Displacement>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Displacement>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Forces>Body Forces>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Species>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Species>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Species>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Turbulence>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Turbulence>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Turbulence>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Velocity>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Velocity>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Velocity>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Velocity>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Volume Fract>Bound Loads>On Elements
Main Menu>Preprocessor>Loads>Define Loads>Apply>Fluid/CFD>Volume Fract>Bound Loads>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Structural>Pressure>On Element Components

Main Menu>Preprocessor>Loads>Define Loads>Apply>Structural>Pressure>On Elements
Main Menu>Preprocessor>Loads>Define Loads>Apply>Structural>Pressure>On Node Components
Main Menu>Preprocessor>Loads>Define Loads>Apply>Structural>Pressure>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Ambient Rad>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Ambient Rad>On Elements

Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Ambient Rad>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Ambient Rad>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Convection>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Convection>On Elements>Uniform
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Convection>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Convection>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Flux>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Flux>On Elements
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Flux>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Flux>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Areas
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Elements
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Keypoints
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Lines
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Nodes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>On Volumes
Main Menu>Preprocessor>Loads>Define Loads>Apply>Thermal>Heat Generat>Uniform Heat Gen
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>TimeInt>On Areas
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>TimeInt>On Keypoints
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>TimeInt>On Nodes
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>Voltage>On Areas
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>Voltage>On Keypoints
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>Voltage>On Lines
Main Menu>Solution>Define Loads>Apply>Electric>Boundary>Voltage>On Nodes
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>AppCharge>On Keypoints
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>AppCharge>On Nodes
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>AppCurrent>On Keypoints
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>AppCurrent>On Nodes
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>ImprCurr>On Keypoints
Main Menu>Solution>Define Loads>Apply>Electric>Excitation>ImprCurr>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Flow>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Flow>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Elements
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Heat Generat>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Heat Generat>Uniform Heat Gen
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Areas
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/ANSYS>Pressure DOF>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Displacement>On Areas
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Displacement>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Displacement>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Displacement>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Forces>Body Forces>On Nodes

Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Areas
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Pressure DOF>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Species>On Areas

Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Species>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Species>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Turbulence>On Areas
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Turbulence>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Turbulence>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Velocity>On Areas
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Velocity>On Keypoints
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Velocity>On Lines
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Velocity>On Nodes
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Volume Fract>Bound Loads>On Elements
Main Menu>Solution>Define Loads>Apply>Fluid/CFD>Volume Fract>Bound Loads>On Lines
Main Menu>Solution>Define Loads>Apply>Structural>Pressure>On Element Components
Main Menu>Solution>Define Loads>Apply>Structural>Pressure>On Elements
Main Menu>Solution>Define Loads>Apply>Structural>Pressure>On Node Components
Main Menu>Solution>Define Loads>Apply>Structural>Pressure>On Nodes
Main Menu>Solution>Define Loads>Apply>Thermal>Ambient Rad>On Areas
Main Menu>Solution>Define Loads>Apply>Thermal>Ambient Rad>On Elements
Main Menu>Solution>Define Loads>Apply>Thermal>Ambient Rad>On Lines
Main Menu>Solution>Define Loads>Apply>Thermal>Ambient Rad>On Nodes
Main Menu>Solution>Define Loads>Apply>Thermal>Convection>On Areas
Main Menu>Solution>Define Loads>Apply>Thermal>Convection>On Elements>Uniform
Main Menu>Solution>Define Loads>Apply>Thermal>Convection>On Lines
Main Menu>Solution>Define Loads>Apply>Thermal>Convection>On Nodes
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Flux>On Areas
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Flux>On Elements
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Flux>On Lines
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Flux>On Nodes
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Areas
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Elements
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Keypoints
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Lines
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Nodes
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>On Volumes
Main Menu>Solution>Define Loads>Apply>Thermal>Heat Generat>Uniform Heat Gen
Utility Menu>Parameters>Array Parameters>Read from File

/UCMD, *Cmd*, *SRNUM*

Assigns a user-defined command name.

APDL: Abbreviations

MP ME ST <> <> <> <> <> <> PP <>

Cmd

User-defined command name. Only the first four characters are significant. Must not conflict with any ANSYS command name or any user "unknown command" macro name.

SRNUM

User subroutine number (1 to 10) programmed for this command. For example, the command **/UCMD**,MYCMD,3 will execute subroutine USER03 whenever the command **MYCMD** is entered. Use a blank command name to disassociate *SRNUM* from its command. For example, **/UCMD**,,3 removes **MYCMD** as a command.

Notes

Assigns a user-defined command name to a user-programmable (system-dependent) subroutine. This feature allows user-defined commands to be programmed into the ANSYS program. Once programmed, this command can be input to the program like other commands, and can also be included in the ANSYS start-up file. See ***ULIB** for another way of defining user commands.

Up to 10 subroutines are available for user-defined commands (USER01 to USER10). Users must have system permission, system access, and knowledge to write, compile, and link the appropriate subprocessors into the ANSYS program at the site where it is to be run. All routines should be written in FORTRAN 77. The USER01 routine is commented and should be listed from the distribution media (system dependent) for more details. Issue **/UCMD**,STAT to list all user-defined command names. Since a user-programmed command is a nonstandard use of the program, the verification of any ANSYS run incorporating these commands is entirely up to the user. In any contact with ANSYS customer support regarding the performance of a custom version of the ANSYS program, you should explicitly state that a user programmable feature has been used. See the *ANSYS Advanced Analysis Techniques Guide* for a general description of user-programmable features and *Guide to ANSYS User Programmable Features* for a detailed description of these features.

This command is valid only at the Begin Level.

Menu Paths

This command cannot be accessed from a menu.

***ULIB**, *Fname*, *Ext*, --

Identifies a macro library file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Fname

File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

Ext

Filename extension (8 character maximum).

--
Unused field

Command Default

No macro library file.

Notes

Identifies a macro library file for the ***USE** command. A library of macros allows blocks of often used ANSYS commands to be stacked and executed from a single file. The macro blocks must be enclosed within block identifier and terminator lines as shown in the example below. If you want to add comment lines to a macro block, you may place them anywhere *within* the macro block. (This includes placing them directly on the lines where the macro block identifier and the macro block terminator appear, as shown in the example.) Do not place comment lines (or any other lines) outside of a macro block.

```
ABC! Any valid alphanumeric name (8 characters maximum)
!  identifying this data block
---! ANSYS data input commands
---
---
/EOF! Terminator for this data block
XYZ! Identify another data block (if desired)
---! ANSYS data input commands
---
---
/EOF! Terminator for this data block
(etc.)
```

The name of the macro library file is identified for reading on the ***ULIB** command. The name of the macro block is identified on the ***USE** command. The commands within the macro block are copied to a temporary file (of the macro block name) during the ***USE** operation and executed as if a macro file of that name had been created by the user. The temporary file is deleted after it has been used. Macro block names should be acceptable filenames (system dependent) and should not match user created macro file names, since the user macro file will be used first (if it exists) before the library file is searched. Macro blocks may be stacked in any order. Branching [***GO** or ***IF**] external to the macro block is not allowed.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Execute Data Block

***USE**, *Name*, *ARG1*, *ARG2*, *ARG3*, *ARG4*, *ARG5*, *ARG6*, *ARG7*, *ARG8*, *ARG9*, *AR10*, *AR11*, *AR12*, *AR13*, *AR14*, *AG15*, *AR16*, *AR17*, *AR18*

Executes a macro file.

APDL: Macro Files

MP ME ST DY <> PR EM <> FL PP ED

Name
Name (32 characters maximum, beginning with a letter) identifying the macro file or a macro block on a macro library file.

ARG1, *ARG2*, *ARG3*, *ARG4*, *ARG5*, *ARG6*, *ARG7*, *ARG8*, *ARG9*, *AR10*, *AR11*, *AR12*, *AR13*, *AR14*, *AG15*, *AR16*, *AR17*, *AR18*
Values passed into the file or block where the parameters ARG1 through ARG9 and AR10 through AR18 are referenced. Values may be numbers, alphanumeric character strings (up to 8 characters enclosed in single quotes), parameters (numeric or character) or parametric expressions. See below for additional details.

Notes

Causes execution of a macro file called *Name*, or, if not found, a macro block "*Name*" on the macro library file [***ULIB**]. Argument values (numeric or character) are passed into the file or block and substituted for local parameters ARG1, ARG2, ..., AR18. The file *Name* may also be executed as an "unknown command" (i.e., without the ***USE** command name) as described below.

A macro is a sequence of ANSYS commands (as many as needed) recorded in a file or in a macro block in a library file (specified with the ***ULIB** command). The file or block is typically executed with the ***USE** command. In addition to command, numerical and alphanumeric data, the macro may include parameters which will be assigned numerical or alphanumerical character values when the macro is used. Use of the macro may be repeated (within a do-loop, for example) with the parameters incremented. A macro is defined within a run by "enclosing" a sequence of data input commands between a ***CREATE** and a ***END** command. The data input commands are passive (not executed) while being written to the macro file. The macro file (without ***CREATE** and ***END**) can also be created external to ANSYS.

Up to 99 specially named scalar parameters called ARG1 to AR99 are *locally* available to each macro. Note that the prefix for the first 9 parameters is "ARG," while the prefix for the last 90 is "AR." A *local* parameter is one which is not affected by, nor does it affect, other parameters, even those of the same name, which are used outside of the macro. The only way a local parameter can affect, or be affected by, parameters outside the macro is if values are passed out of, or into, the macro by an argument list. Parameters ARG1 through AR18 can have their values (numeric or character) passed via the argument list on the ***USE** command (ARG1 through AR19 can be passed as arguments on the "unknown command" macro). Parameters AR19 through AR99 (AR20 through AR99 in the "unknown command" macro) are available solely for use within the macro; they cannot be passed via an argument list. Local parameters are available to do-loops and to **/INPUT** files processed within the macro. In addition to an ARG1--AR99 set for each macro, another ARG1--AR99 set is available external to all macros, local to "non-macro" space.

A macro is exited after its last line is executed. Macros may be nested (such as a ***USE** or an "unknown command" within a macro). Each nested macro has its own set of 99 local parameters. Only one set of local parameters can be active at a time and that is the set corresponding to the macro currently being executed or to the set external to all macros (if any). When a nested macro completes execution, the previous set of local parameters once again becomes available. Use ***STATUS**,ARGX to view current macro parameter values.

An alternate way of executing a macro file is via the "unknown command" route. If a command unknown to the ANSYS program is entered, a search for a file of that name (plus a **.MAC** suffix) is made. If the file exists, it is executed, if not, the "unknown command" message is output. Thus, users can write their own commands in terms of other ANSYS commands. The procedure is similar to issuing the ***USE** command with the unknown command in the *Name* field. For example, the command **CMD**,10,20,30 is internally similar to ***USE**,CMD,10,20,30. The macro file named **CMD.MAC** will be executed with the three parameters. The ***USE** macro description also applies to the "unknown command" macro, except that various directories are searched and a suffix (**.MAC**) is assumed. Also, a macro library file is not searched.

A three-level directory search for the "unknown command" macro file may be available (see the *ANSYS Operations Guide*). The search order may be: 1) a high-level system directory, 2) the login directory, and 3) the local (working) directory. Use the **/PSEARCH** command to change the directory search path. For an "unknown command" **CMD**, the first file named **CMD.MAC** found to exist in the search order will be executed. The command may be input

as upper or lower case, however, it is converted to upper case before the file name search occurs. On systems that uniquely support both upper and lower case file names, the file with the matching lower case name will be used if it exists, otherwise, the file with the matching upper case name will be used. All macro files placed in the **apdl** directory must be upper case.

Note, since undocumented commands exist in the ANSYS program, the user should issue the command intended for the macro file name to be sure the "unknown command" message is output in the processor where it's to be used. If the macro is to be used in other processors, the other processors must also be checked.

This command is valid in any processor.

Menu Paths

Utility Menu>Macro>Execute Data Block

---

*VABS, *KABSR*, *KABS1*, *KABS2*, *KABS3*

Applies the absolute value function to array parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

KABSR

Absolute value of results parameter:

0 --
Do not take absolute value of results parameter (ParR).

1 --
Take absolute value.

KABS1

Absolute value of first parameter:

0 --
Do not take absolute value of first parameter (Par1 or ParI).

1 --
Take absolute value.

KABS2

Absolute value of second parameter:

0 --
Do not take absolute value of second parameter (Par2 or ParJ).

1 --
Take absolute value.

KABS3

Absolute value of third parameter:

0 --
Do not take absolute value of third parameter (Par3 or ParK).

1 --
Take absolute value.

Command Default

Do not use absolute values.

Notes

Applies an absolute value to parameters used in certain ***V*XX** and ***M*XX** operations. Typical absolute value applications are of the form:

ParR = |f(|Par1|)|

or

ParR = |(|Par1| o |Par2|)|

The absolute values are applied to each input parameter value before the operation and to the result value after the operation. Absolute values are applied before the scale factors so that negative scale factors may be used. The absolute value settings are *reset* to the default (no absolute value) after each ***V*XX** or ***M*XX** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

***VCOL**, *NCOL1*, *NCOL2*

Specifies the number of columns in matrix operations.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

NCOL1
Number of columns to be used for Par1 with ***M*XX** operations. Defaults to whatever is needed to fill the result array.

NCOL2
Number of columns to be used for Par2 with ***M*XX** operations. Defaults to whatever is needed to fill the result array.

Command Default

Fill all locations of the result array from the specified starting location.

Notes

Specifies the number of columns to be used in array parameter matrix operations. The size of the submatrix used is determined from the upper left starting array element (defined on the operation command) to the lower right

array element (defined by the number of columns on this command and the number of rows on the ***VLEN** command).

The default *NCOL* is calculated from the maximum number of columns of the result array (the ***DIM** column dimension) minus the starting location + 1. For example, ***DIM**,R,,1,10 and a starting location of R(1,7) gives a default of 4 columns (starting with R(1,7), R(1,8), R(1,9), and R(1,10)). Repeat operations automatically terminate at the last column of the result array. Existing values in the rows and columns of the results matrix remain unchanged where not overwritten by the requested input or operation values.

The column control settings are *reset* to the defaults after each ***M*XX*** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

***VCUM**, *KEY*

Allows array parameter results to add to existing results.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

KEY

Accumulation key:

0 --

Overwrite results.

1 --

Add results to the current value of the results parameter.

Command Default

Overwrite results.

Notes

Allows results from certain ***V*XX*** and ***M*XX*** operations to overwrite or add to existing results. The cumulative operation is of the form:

ParR = ParR + ParR(Previous)

The cumulative setting is *reset* to the default (overwrite) after each ***V*XX*** or ***M*XX*** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

***VEDIT**, *Par*

Allows numerical array parameters to be graphically edited.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par

Name of the array parameter to be edited.

Notes

Invokes a graphical editing system that displays array parameter values in matrix form, and allows the use of the mouse to edit individual values. The starting array subscripts must be defined, such as ***VEDIT**,A(4,6,1), to indicate the section of the array to be edited. The array section starts at the specified array element and continues to the maximum extent of the array parameter. Row and column index values may be set or changed in any plane, and those values will be applied to all planes. The menu system must be on [**/MENU**] when this command is issued. Graphical editing is not available for character array parameters. The ***VEDIT** command can not be used in a macro or other secondary input file.

This command is not applicable to 4- or 5-D arrays.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Parameters>Define/Edit

***VFACT**, *FACTR*, *FACT1*, *FACT2*, *FACT3*

Applies a scale factor to array parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

FACTR

Scale factor applied to results (ParR) parameter. Defaults to 1.0.

FACT1

Scale factor applied to first parameter (Par1 or ParI). Defaults to 1.0.

FACT2

Scale factor applied to second parameter (Par2 or ParJ). Defaults to 1.0.

FACT3

Scale factor applied to third parameter (Par3 or ParK). Defaults to 1.0.

Command Default

Use 1.0 for all scale factors.

Notes

Applies a scale factor to parameters used in certain ***V*XX** and ***M*XX** operations. Typical scale factor applications are of the form:

ParR = *FACTR**f(*FACT1**Par1)

or

ParR = *FACTR**((*FACT1**Par1) o (*FACT2**Par2))

The factors are applied to each input parameter value before the operation and to the result value after the operation. The scale factor settings are *reset* to the default (1.0) after each ***V*XX** or ***M*XX** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

---

*VFILL, *ParR*, *Func*, *CON1*, *CON2*, *CON3*, *CON4*, *CON5*, *CON6*, *CON7*, *CON8*, *CON9*, *CON10*

Fills an array parameter.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the resulting numeric array parameter vector. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Func
Fill function:

DATA --
Assign specified values *CON1*, *CON2*, etc. to successive array elements. Up to 10 assignments may be made at a time. Any *CON* values after a blank *CON* value are ignored.

RAMP --
Assign ramp function values: *CON1*+((n-1)**CON2*) , where n is the loop number [***VLEN**]. To specify a constant function (no ramp), set *CON2* to zero.

RAND --
Assign random number values based on a uniform distribution: RAND(*CON1*,*CON2*), where *CON1* is the lower bound (defaults to 0.0) and *CON2* is the upper bound (defaults to 1.0).

GDIS --
Assign random sample of Gaussian distributions: GDIS(*CON1*,*CON2*), where *CON1* is the mean (defaults to 0.0) and *CON2* is the standard deviation (defaults to 1.0).

TRIA --
Assigns random number values based on a triangular distribution: TRIA(*CON1*,*CON2*,*CON3*), where *CON1* is the lower bound (defaults to 0.0), *CON2* is the location of the peak value (CON1 $\leq$ CON2 $\leq$ *CON3*; *CON2*

defaults to 0 if *CON1* ≤ 0 ≤ *CON3*, *CON1* if 0 ≤ *CON1*, or *CON3* if *CON3* ≤ 0), and *CON3* is the upper bound (defaults to 1.0 + *CON1* if *CON1* ≥ 0 or 0.0 if *CON1* ≤ 0).

BETA --
Assigns random number values based on a beta distribution: BETA(*CON1*,*CON2*,*CON3*,*CON4*), where *CON1* is the lower bound (defaults to 0.0), *CON2* is the upper bound (defaults to 1.0+*CON1* if *CON1* ≥ 0 or 0.0 if *CON1* ≤ 0), and *CON3* and *CON4* are the alpha and beta parameters, respectively, of the beta function. Alpha and beta must both be positive; they default to 1.0.

GAMM --
Assigns random number values based on a gamma distribution: GAMM(*CON1*,*CON2*,*CON3*), where *CON1* is the lower bound (defaults to 0.0), and *CON2* and *CON3* are the alpha and beta parameters. respectively, of the gamma function. Alpha and beta must both be positive; they default to 1.0.

CON1, *CON2*, *CON3*, *CON4*, *CON5*, *CON6*, *CON7*, *CON8*, *CON9*, *CON10*
Constants used with above functions.

Notes

Operates on input data and produces one output array parameter vector according to:

ParR = f(*CON1*, *CON2*, ...)

where the functions (f) are described above. A starting array element number must be defined for the result array parameter vector. Operations use successive array elements [***VLEN**, ***VMASK**] with the default being all successive elements. For example, ***VFILL**,A(1),RAMP,1,10 assigns A(1) = 1.0, A(2) = 11.0, A(3) = 21.0, etc. ***VFILL**,B(5,1),DATA,1.5,3.0 assigns B(5,1) = 1.5 and B(6,1) = 3.0. Absolute values and scale factors may be applied to the result parameter [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Parameters>Fill

***VFUN**, *ParR*, *Func*, *Par1*, *CON1*, *CON2*, *CON3*

Performs a function on a single array parameter.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the resulting numeric array parameter vector. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Func
Function to be performed:

ACOS --
Arccosine: ACOS(*Par1*).

ASIN --
Arcsine: ASIN(*Par1*).

ASORT --
Par1 is sorted in ascending order. ***VCOL**, ***VMASK**, ***VCUM**, and ***VLEN**,,NINC do not apply. ***VLEN**,NROW does apply.

ATAN --
Arctangent: ATAN(*Par1*).

COMP --
Compress: Selectively compresses data set. "True" (***VMASK**) values of *Par1* (or row positions to be considered according to the *NINC* value on the ***VLEN** command) are written in compressed form to *ParR*, starting at the specified position.

COPY --
Copy: *Par1* copied to *ParR*.

COS --
Cosine: COS(*Par1*).

COSH --
Hyperbolic cosine: COSH(*Par1*).

DIRCOS --
Direction cosines of the principal stresses (*n*X9). *Par1* contains the *n*X6 component stresses for the *n* locations of the calculations.

DSORT --
Par1 is sorted in descending order. ***VCOL**, ***VMASK**, ***VCUM**, and ***VLEN**,,NINC do not apply. ***VLEN**,NROW does apply.

EULER --
Euler angles of the principal stresses (*n*X3). *Par1* contains the *n*X6 component stresses for the *n* locations of the calculations.

EXP --
Exponential: EXP(*Par1*).

EXPA --
Expand: Reverse of the COMP function. All elements of *Par1* (starting at the position specified) are written in expanded form to corresponding "true" (***VMASK**) positions (or row positions to be considered according to the *NINC* value on the ***VLEN** command) of *ParR*.

LOG --
Natural logarithm: LOG(*Par1*).

LOG10 --
Common logarithm: LOG10(*Par1*).

NINT --
Nearest integer: 2.783 becomes 3.0, -1.75 becomes -2.0.

NOT --
Logical complement: values $\leq$ 0.0 (false) become 1.0 (true). Values > 0.0 (true) become 0.0 (false).

PWR --
Power function: *Par1***CON1*. Exponentiation of any negative number in the vector *Par1* to a non-integer power is performed by exponentiating the positive number and prepending the minus sign. For example, -4**2.3 is -(4**2.3).

SIN --
Sine: SIN(*Par1*).

SINH --
Hyperbolic sine: SINH(*Par1*).

SQRT --
Square root: SQRT(*Par1*).

TAN --
Tangent: TAN(*Par1*).

TANH --
Hyperbolic tangent: TANH(*Par1*).

TANG --
Tangent to a path at a point: the slope at a point is determined by linear interpolation half way between the previous and next points. Points are assumed to be in the global Cartesian coordinate system. Path points are specified in array *Par1* (having 3 consecutive columns of data, with the columns containing the x, y, and z coordinate locations, respectively, of the points). Only the starting row index and the column index for the x coordinates are specified, such as A(1,1). The y and z coordinates of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3). The tangent result, *ParR*, must also have 3 consecutive columns of data and will contain the tangent direction vector (normalized to 1.0); such as 1,0,0 for an x-direction vector.

NORM --
Normal to a path and an input vector at a point: determined from the cross-product of the calculated tangent vector (see TANG) and the input direction vector (with the i, j, and k components input as *CON1*, *CON2*, and *CON3*). Points are assumed to be in the global Cartesian coordinate system. Path points are specified in array *Par1* (having 3 consecutive columns of data, with the columns containing the x, y, and z coordinate locations, respectively, of the points). Only the starting row index and the column index for the x coordinates are specified, such as A(1,1). The y and z coordinates of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3). The normal result, *ParR*, must also have 3 consecutive columns of data and will contain the normal direction vector (normalized to 1.0); such as 1,0,0 for an x-direction vector.

LOCAL --
Transforms global Cartesian coordinates of a point to the coordinates of a specified system: points to be transformed are specified in array *Par1* (having 3 consecutive columns of data, with the columns containing the x, y, and z global Cartesian coordinate locations, respectively, of the points). Only the starting row index and the column index for the x coordinates are specified, such as A(1,1). The y and z coordinates of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3). Results are transformed to coordinate system *CON1* (which may be any valid coordinate system number, such as 1,2,11,12, etc.). The transformed result, *ParR*, must also have 3 consecutive columns of data and will contain the corresponding transformed coordinate locations.

GLOBAL --
Transforms specified coordinates of a point to global Cartesian coordinates: points to be transformed are specified in array *Par1* (having 3 consecutive columns of data, with the columns containing the local coordinate locations (x, y, z or r, θ, z or etc.) of the points). Only the starting row index and the column index for the x coordinates are specified, such as A(1,1). The y and z coordinates (or θ and z, or etc.) of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3). Local coordinate locations are assumed to be in coordinate system *CON1* (which may be any valid coordinate system number, such as 1,2,11,12, etc.). The transformed result, *ParR*, must also have 3 consecutive columns of data, with the columns containing the global Cartesian x, y, and z coordinate locations, respectively.

Par1
Array parameter vector in the operation.

CON1, *CON2*, *CON3*
Constants (used only with the PWR, NORM, LOCAL, and GLOBAL functions).

Notes

Operates on one input array parameter vector and produces one output array parameter vector according to:

ParR = f(*Par1*)

where the functions (f) are described below. Functions are based on the standard FORTRAN definitions where possible. Out-of-range function results (or results with exponents whose magnitudes are approximately greater than 32 or less than -32) produce a zero value. Input and output for angular functions may be radians (default) or degrees [***AFUN**]. *ParR* may be the same as *Par1*. Starting array element numbers must be defined for each array parameter vector. For example, ***VFUN**,A(1),SQRT,B(5) takes the square root of the fifth element of B and stores the result in the first element of A. Operations continue on successive array elements [***VLEN**, ***VMASK**] with the default being all successive elements. Absolute values and scale factors may be applied to all parameters [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. Skipping array elements via ***VMASK** or ***VLEN** for the TANG and NORM functions skips only the writing of the results (skipped array element data are used in all calculations). See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Vector Functions

***VGET**, *ParR*, *Entity*, *ENTNUM*, *Item1*, *IT1NUM*, *Item2*, *IT2NUM*, *KLOOP*

Retrieves values and stores them into an array parameter.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the resulting vector array parameter. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Entity
Entity keyword. Valid keywords are NODE, ELEM, KP, LINE, AREA, VOLU, etc. as shown for *Entity* = in the table below.

ENTNUM
The number of the entity (as shown for *ENTNUM* = in the table below).

Item1
The name of a particular item for the given entity. Valid items are as shown in the *Item1* columns of the table below.

IT1NUM
The number (or label) for the specified *Item1* (if any). Valid *IT1NUM* values are as shown in the *IT1NUM* columns of the table below. Some *Item1* labels do not require an *IT1NUM* value.

Item2, *IT2NUM*
A second set of item labels and numbers to further qualify the item for which data is to be retrieved. Most items do not require this level of information.

KLOOP
Field to be looped on:

0 or 2 --
Loop on the *ENTNUM* field (default).

3 --
Loop on the *Item1* field.

4 --
Loop on the *IT1NUM* field. Successive items are as shown with *IT1NUM*.

5 --
Loop on the *Item2* field.

6 --
Loop on the *IT2NUM* field. Successive items are as shown with *IT2NUM*.

Notes

Retrieves values for specified items and stores the values in an output vector of a user-named array parameter according to:

ParR = f(*Entity*, *ENTNUM*, *Item1*, *IT1NUM*, *Item2*, *IT2NUM*)

where (f) is the ***GET** function; *Entity*, *Item1*, and *Item2* are keywords; and *ENTNUM*, *IT1NUM*, and *IT2NUM* are numbers or labels corresponding to the keywords. A starting array location number must be defined for the result array parameter. Looping continues over successive entity numbers (*ENTNUM*) for the *KLOOP* default. For example, ***VGET**,A(1),ELEM,5,CENT,X returns the centroid x-location of element 5 and stores the result in the first location of A. Retrieving continues with element 6, 7, 8, etc. until successive array locations [***VLEN**, ***VMASK**] are filled. Absolute values and scale factors may be applied to the result parameter [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. See the ***VOPER** command for general details. Results can be put back into an analysis by writing a file of the desired input commands with the ***VWRITE** command. See also the ***VPUT** command.

Both ***GET** and ***VGET** retrieve information from the active data stored in memory. The database is often the source, and sometimes the information is retrieved from common memory blocks that ANSYS uses to manipulate information. Although POST1 and POST26 operations use a ***.rst** file, GET data is accessed from the database or from the common blocks. Get operations do not access the ***.rst** file directly.

The ***VGET** command retrieves both the unprocessed real and the imaginary parts (original and duplicate sector nodes and elements) of a cyclic symmetry solution.

This command is valid in any processor.

*VGET - PREP7 Items

PREP7 Items

Entity = NODE, *ENTNUM* = *n* (node number)

Item1	IT1NUM	Description

PREP7 Items

LOC	X, Y, Z	X, Y, or Z location in the active coordinate system.
ANG	XY, YZ, ZX	THXY, THYZ, THZX rotation angle.
NSEL		Select status of node *n* (-1 - unselected, 0 - undefined, 1 - selected).

Entity = ELEM, *ENTNUM* = *n* (element number)

Item1	IT1NUM	Description
NODE	1,2,--20	Node number at position 1,2,--20 of element *n*.
CENT	X, Y, Z	Centroid X, Y, or Z location (based on shape function) in the active coordinate system.
ADJ	1, 2, -- 6	Number of element adjacent to face 1, 2, -- 6.
ATTR	name	Number assigned to attribute *name* = MAT, TYPE, REAL, ESYS, ENAM, or SECN).
GEOM		Characteristic element geometry. Length of line element (straight line between ends), area of area element, or volume of volume element. Issuing ***VGET** for an element returns a signed value. To always get a positive value, issue ***VABS**,1 just prior to issuing ***VGET**,par(n),ELEM,x,GEOM.
ESEL		Select status of element *n* (-1 - unselected, 0 - undefined, 1 - selected).
SHPAR	*Test*	Element shape test result for selected element *n*, where *Test*=ANGD (SHELL28 corner angle deviation), ASPE (aspect ratio), JACR (Jacobian ratio), MAXA (maximum corner angle), PARA (deviation from parallelism of opposite edges), or WARP (warping factor).

Entity = KP, *ENTNUM* = *n* (keypoint number)

Item1	IT1NUM	Description
LOC	X, Y, Z	X, Y, or Z location in the active coordinate system.
ATTR	name	Number assigned to attribute (*name* = MAT, TYPE, REAL, ESYS, NODE or ELEM).
DIV		Divisions (element size setting) from **KESIZE** command.
KSEL		Select status of keypoint *n* (-1 - unselected, 0 - undefined, 1 - selected).

Entity = LINE, *ENTNUM* = *n* (line number)

Item1	IT1NUM	Description
KP	1,2	Keypoint number at position 1 or 2.
ATTR	name	Number assigned to attribute (*name* = MAT, TYPE, REAL, ESYS, NNOD, NELM, or NDIV). NNOD = number of nodes, NELM = number of elements, NDIV = number of divisions.
LENG		Length.
LSEL		Select status of line *n* (-1 - unselected, 0 - undefined, 1 - selected).

Entity = AREA, *ENTNUM* = *n* (area number)

Item1	IT1NUM	Description
LOOP	1,2,-- *I*	Loop number. Must be input if LINE number is to be retrieved.
Item2	IT2NUM	Description
LINE	1, 2, -- *p*	Line number at position 1, 2, --- *p*.
ATTR	name	Number assigned to attribute (*name* = MAT, TYPE, REAL, ESYS, NNOD, or NELM). NNOD = number of nodes, NELM = number of elements.
AREA		Area (after last **ASUM**).
ASEL		Select status of area *n* (-1 - unselected, 0 - undefined, 1 - selected).

PREP7 Items

Entity = VOLU, *ENTNUM* = *n* (volume number)

Item1	IT1NUM	Description
SHELL	1,2,--*l*	Shell number. Must be input if AREA number is to be retrieved.
Item2	IT2NUM	Description
AREA	1, 2, -- *p*	Area number at position 1, 2, --- *p*.
ATTR	name	Number assigned to attribute (*name* = MAT, TYPE, REAL, ESYS, NNOD, or NELM). NNOD = number of nodes, NELM = number of elements.
VOLU		Volume (after last **VSUM**).
VSEL		Select status of volume *n* (-1 - unselected, 0 - undefined, 1 - selected).

Entity = CDSY, *ENTNUM* = *n* (coordinate system number)

Item1	IT1NUM	Description
LOC	X, Y, Z	X, Y, or Z origin location (global Cartesian coordinate).
ANG	XY, YZ, ZX	THXY, THYZ, or THZX rotation angle (°) relative to the global Cartesian coordinate system.
ATTR	name	Number assigned to attribute (*name* = KCS, KTHET, KPHI, PAR1, or PAR2). A -1.0 is returned for KCS if coordinate system is undefined).

Entity = RCON, *ENTNUM* = *n* (real constant set number)

Item1	IT1NUM	Description
CONST	1,2,--*m*	Real constant value for constant 1,2,--- *m*.

Entity = TLAB, *ENTNUM* = *n* (*Tlab* is the data table label: BKIN, MKIN, MISO, etc. as described on the **TB** command. *n* is the material number.)

Item1	IT1NUM	Description
TEMP	val	Temperature value (if any) at which to retrieve table data.
Item2	IT2NUM	Description
CONST	*num*	Constant number whose value is to be retrieved (see Data Tables - Implicit Analysis in the *ANSYS Elements Reference*). For constants input as X, Y points, the constant numbers are consecutive with the X constants being the odd numbers, beginning with one.

*VGET - POST1 Items

***Entity* = NODE, *ENTNUM* = *n* (node number)**

Vector items are in the active results coordinate system unless otherwise specified.

Item1	IT1NUM	Description
Valid labels for nodal degree of freedom results are:		
U	X, Y, Z	X, Y, or Z structural displacement.
ROT	X, Y, Z	X, Y, or Z structural rotation.
TEMP		Temperature. For SHELL131 and SHELL132 elements with KEYOPT(3) = 0 or 1, use TBOT, TE2, TE3, . . ., TTOP instead of TEMP. Alternative get functions: TEMP(N), TBOT(N), TE2(N), etc.
PRES		Pressure.
VOLT		Electric potential.
MAG		Magnetic scalar potential.
V	X, Y, Z	X, Y, or Z fluid velocity.

Entity **= NODE, *ENTNUM* = *n* (node number)**

Vector items are in the active results coordinate system unless otherwise specified.

Item1	IT1NUM	Description
A	X, Y, Z	X, Y, or Z magnetic vector potential.
CURR		Current.
EMF		Electromotive force drop.
ENKE		Turbulent kinetic energy (FLOTRAN).
ENDS		Turbulent energy dissipation (FLOTRAN).

Valid labels for element nodal results are:

Item1	IT1NUM	Description
S	X, Y, Z, XY, YZ, XZ	Component stress.
"	1,2,3	Principal stress.
"	INT, EQV	Stress intensity or equivalent stress.
EPTO	X, Y, Z, XY, YZ, XZ	Component total strain (EPEL + EPPL + EPCR).
"	1, 2, 3	Principal total strain.
"	INT, EQV	Total strain intensity or total equivalent strain.
EPEL	X, Y, Z, XY, YZ, XZ	Component elastic strain.
"	1, 2, 3	Principal elastic strain.
"	INT, EQV	Elastic strain intensity or elastic equivalent strain.
EPPL	X, Y, Z, XY, YZ, XZ	Component plastic strain.
"	1, 2, 3	Principal plastic strain.
"	INT, EQV	Plastic strain intensity or plastic equivalent strain.
EPCR	X, Y, Z, XY, YZ, XZ	Component creep strain.
"	1, 2, 3	Principal creep strain.
"	INT, EQV	Creep strain intensity or creep equivalent strain.
EPTH	X, Y, Z, XY, YZ, XZ	Component thermal strain.
"	1, 2, 3	Principal thermal strain.
"	INT, EQV	Thermal strain intensity or thermal equivalent strain.
EPSW		Swelling strain.
NL	SEPL	Equivalent stress (from stress-strain curve).
"	SRAT	Stress state ratio.
"	HPRES	Hydrostatic pressure.
"	EPEQ	Accumulated equivalent plastic strain.
"	PSV	Plastic state variable.
"	PLWK	Plastic work/volume.
HS	X, Y, Z	Component magnetic field intensity from current sources (in the global Cartesian coordinate system).
BFE	TEMP	Body temperatures (calculated from applied temperatures) as used in solution.
TG	X, Y, Z, SUM	Component thermal gradient and sum.
TF	X, Y, Z, SUM	Component thermal flux and sum.
PG	X, Y, Z, SUM	Component pressure gradient and sum.
EF	X, Y, Z, SUM	Component electric field and sum.

***Entity* = NODE, *ENTNUM* = *n* (node number)**

Vector items are in the active results coordinate system unless otherwise specified.

Item1	IT1NUM	Description
D	X, Y, Z, SUM	Component electric flux density and sum.
H	X, Y, Z, SUM	Component magnetic field intensity and sum.
B	X, Y, Z, SUM	Component magnetic flux density and sum.
FMAG	X, Y, Z, SUM	Component magnetic force and sum.

Valid labels for FLOTRAN nodal results are:

Item1	IT1NUM	Description
TTOT		Total temperature.
HFLU		Heat flux.
HFLM		Heat transfer (film) coefficient.
COND		Fluid laminar conductivity.
PCOE		Pressure coefficient.
PTOT		Total (stagnation) pressure.
MACH		Mach number.
STRM		Stream function. (2-D applications only.)
DENS		Fluid density.
VISC		Fluid laminar viscosity.
EVIS		Fluid effective viscosity.
ECON		Fluid effective conductivity.
YPLU		Y+, a turbulent law of the wall parameter.
TAUW		Shear stress at the wall.

Entity = ELEM, *ENTNUM* = *n* (element number)

Valid labels for element results are:

ETAB	Lab	Any user-defined element table label (see **ETABLE** command).

Menu Paths

Utility Menu>Parameters>Get Array Data

***VITRP**, *ParR*, *ParT*, *ParI*, *ParJ*, *ParK*

Forms an array parameter by interpolation of a table.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR

The name of the resulting array parameter. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

ParT

The name of the TABLE array parameter. The parameter must exist as a dimensioned array of type TABLE [***DIM**].

ParI
Array parameter vector of I (row) index values for interpolation in *ParT*.

ParJ
Array parameter vector of J (column) index values for interpolation in *ParT* (which must be at least 2-D).

ParK
Array parameter vector of K (depth) index values for interpolation in *ParT* (which must be 3-D).

Notes

Forms an array parameter (of type ARRAY) by interpolating values of an array parameter (of type TABLE) at specified table index locations according to:

ParR = f(*ParT*, *Parl*, *ParJ*, *ParK*)

where *ParT* is the type TABLE array parameter, and *ParI*, *ParJ*, *ParK* are the type ARRAY array parameter vectors of index values for interpolation in *ParT*. See the ***DIM** command for TABLE and ARRAY declaration types. Linear interpolation is used. Starting array element numbers must be defined for each array parameter. The starting array element number for the TABLE array (*ParT*) is not used (but a value must be input). For example, ***VIT-RP**,R(5),TAB(1,1),X(2),Y(4) uses the second element of X and the fourth element of Y as index values (row and column) for a 2-D interpolation in TAB and stores the result in the fifth element of R. Operations continue on successive array elements [***VLEN**, ***VMASK**] with the default being all successive elements. Absolute values and scale factors may be applied to the result parameter [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Vector Interpolate

***VLEN**, *NROW*, *NINC*

Specifies the number of rows to be used in array parameter operations.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

NROW
Number of rows to be used with the ***V*XX** or ***M*XX** operations. Defaults to the number of rows needed to fill the result array.

NINC
Perform the operation on every *NINC* row (defaults to 1).

Command Default

Fill all locations of the result array from the specified starting location.

Notes

Specifies the number of rows to be used in array parameter operations. The size of the submatrix used is determined from the upper left starting array element (defined on the operation command) to the lower right array element (defined by the number of rows on this command and the number of columns on the ***VCOL** command). *NINC* allows skipping row operations for some operation commands. Skipped rows are included in the row count. The starting row number must be defined on the operation command for each parameter read and for the result written.

The default *NROW* is calculated from the maximum number of rows of the result array (the ***DIM** row dimension) minus the starting location + 1. For example, ***DIM**,R,,10 and a starting location of R(7) gives a default of 4 loops (filling R(7), R(8), R(9), and R(10)). Repeat operations automatically terminate at the last row of the result array. Existing values in the rows and columns of the results matrix remain unchanged where not overwritten by the requested input or operation values.

The stride (*NINC*) allows operations to be performed at regular intervals. It has no effect on the total number of row operations. Skipped operations retain the previous result. For example, ***DIM**,R,,6, with a starting location of R(1), *NROW* = 10, and *NINC* = 2 calculates values for locations R(1), R(3), and R(5) and retains values for locations R(2), R(4), and R(6). A more general skip control may be done by masking [***VMASK**]. The row control settings are *reset* to the defaults after each ***V*XX*** or ***M*XX*** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

---

*VMASK, *Par*

Specifies an array parameter as a masking vector.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par

Name of the mask parameter. The starting subscript must also be specified.

Command Default

No mask parameter specified (use true for all operations).

Notes

Specifies the name of the parameter whose values are to be checked for each resulting row operation. The mask vector usually contains only 0 (for false) and 1 (for true) values. For each row operation the corresponding mask vector value is checked. A true value allows the operation to be done. A false value skips the operation (and retains the previous results). A mask vector can be created from direct input, such as M(1) = 1,0,0,1,1,0,1; or from the DATA function of the ***VFILL** command. The NOT function of the ***VFUN** command can be used to reverse the logical sense of the mask vector. The logical compare operations (LT, LE, EQ, NE, GE, and GT) of the ***VOPER** command also produce a mask vector by operating on two other vectors. Any numeric vector can be used as a mask vector since the actual interpretation assumes values less than 0.0 are 0.0 (false) and values greater than

0.0 are 1.0 (true). If the mask vector is not specified (or has fewer values than the result vector), true (1.0) values are assumed for the unspecified values. Another skip control may be input with *NINC* on the ***VLEN** command. If both are present, operations occur only when both are true. The mask setting is *reset* to the default (no mask) after each ***V*XX*** or ***M*XX*** operation. Use ***VSTAT** to list settings.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Operation Settings

***VOPER**, *ParR*, *Par1*, *Oper*, *Par2*, *CON1*, *CON2*

Operates on two array parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the resulting array parameter vector. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**].

Par1
First array parameter vector in the operation. May also be a scalar parameter or a literal constant.

Oper
Operations:

ADD--
Addition: *Par1*+*Par2*.

SUB --
Subtraction: *Par1*-*Par2*.

MULT --
Multiplication: *Par1***Par2*.

DIV --
Division: *Par1*/*Par2* (a divide by zero results in a value of zero).

MIN --
Minimum: minimum of *Par1* and *Par2*.

MAX --
Maximum: maximum of *Par1* and *Par2*.

LT --
Less than comparison: *Par1*<*Par2* gives 1.0 if true, 0.0 if false.

LE --
Less than or equal comparison: *Par1* $\leq$ *Par2* gives 1.0 if true, 0.0 if false.

EQ --
Equal comparison: *Par1* = *Par2* gives 1.0 if true, 0.0 if false.

NE --
Not equal comparison: *Par1* $\neq$ *Par2* gives 1.0 if true, 0.0 if false.

GE --

Greater than or equal comparison: *Par1* $\geq$ Par2 gives 1.0 if true, 0.0 if false.

GT --

Greater than comparison: *Par1*>*Par2* gives 1.0 if true, 0.0 if false.

DER1 --

First derivative: d(*Par1*)/d(*Par2*). The derivative at a point is determined over points half way between the previous and next points (by linear interpolation). *Par1* must be a function (a unique *Par1* value for each *Par2* value) and *Par2* must be in ascending order.

DER2 --

Second derivative: $d^2(Par1)/d(Par2)^2$. See also DER1.

INT1 --

Single integral: $\int Par1\, d(Par2)$, where *CON1* is the integration constant. The integral at a point is determined by using the single integration procedure described in the *ANSYS, Inc. Theory Reference*.

INT2 --

Double integral: $\int\int Par1\, d(Par2)$, where *CON1* is the integration constant of the first integral and *CON2* is the integration constant of the second integral. If *Par1* contains acceleration data, *CON1* is the initial velocity and *CON2* is the initial displacement. See also INT1.

DOT --

Dot product: *Par1* . *Par2*. *Par1* and *Par2* must each have three consecutive columns of data, with the columns containing the i, j, and k vector components, respectively. Only the starting row index and the column index for the i components are specified for *Par1* and *Par2*, such as A(1,1). The j and k components of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3).

CROSS --

Cross product: *Par1* x *Par2*. *Par1*, *Par2*, and *ParR* must each have 3 components, respectively. Only the starting row index and the column index for the i components are specified for *Par1*, *Par2*, and *ParR*, such as A(1,1). The j and k components of the vector are assumed to begin in the corresponding next columns, such as A(1,2) and A(1,3).

GATH --

Gather: For a vector of position numbers, *Par2*, copy the value of *Par1* at each position number to *ParR*. Example: for *Par1* = 10,20,30,40 and *Par2* = 2,4,1; *ParR* = 20,40,10.

SCAT --

Scatter: Opposite of GATH operation. For a vector of position numbers, *Par2*, copy the value of *Par1* to that position number in *ParR*. Example: for *Par1* = 10,20,30,40,50 and *Par2* = 2,1,0,5,3; *ParR* = 20,10,50,0,40.

Par2

Second array parameter vector in the operation. May also be a scalar parameter or a literal constant.

CON1

First constant (used only with the INT1 and INT2 operations).

CON2

Second constant (used only with the INT2 operation).

Notes

Operates on two input array parameter vectors and produces one output array parameter vector according to:

ParR = *Par1* o *Par2*

where the operations (o) are described below. *ParR* may be the same as *Par1* or *Par2*. Absolute values and scale factors may be applied to all parameters [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. Starting array element numbers must be defined for each array parameter vector, such as ***VOPER**,A(1),B(5),ADD,C(3) which adds the third element of C to the fifth element of B and stores the result in the first element of A. Operations continue on successive array elements [***VLEN**, ***VMASK**] with the default being all successive elements. Skipping array elements via ***VMASK** or ***VLEN** for the DER_ and INT_ functions skips only the writing of the results (skipped array element data are used in all calculations).

Parameter functions and operations are available to operate on a scalar parameter or a single element of an array parameter, such as SQRT(B) or SQRT(A(4)). See the ***SET** command for details. Operations on a sequence of array elements can be done by repeating the desired function or operation in a do-loop [***DO**]. The vector operations within the ANSYS program (***V*XX*** commands) are internally programmed do-loops that conveniently perform the indicated operation over a sequence of array elements. If the array is multidimensional, only the first subscript is incremented in the do-loop, that is, the operation repeats in column vector fashion "down" the array. For example, for A(1,5), A(2,5), A(3,5), etc. The starting location of the row index must be defined for each parameter read and for the result written.

The default number of loops is from the starting result location to the last result location and can be altered with the ***VLEN** command. A logical mask vector may be defined to control at which locations the operations are to be skipped [***VMASK**]. The default is to skip no locations. Repeat operations automatically terminate at the last array element of the result array column if the number of loops is undefined or if it exceeds the last result array element. Zeroes are used in operations for values read beyond the last array element of an input array column. Existing values in the rows and columns of the results matrix remain unchanged where not changed by the requested operation values. The result array column may be the same as the input array column since results in progress are stored in a temporary array until being moved to the results array at the end of the operation. Results may be overwritten or accumulated with the existing results [***VCUM**]. The default is to overwrite results. The absolute value may be used for each parameter read or written [***VABS**]. A scale factor (defaulting to 1.0) is also applied to each parameter read and written [***VFACT**].

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Vector Operations

***VPLOT**, *ParX*, *ParY*, *Y2*, *Y3*, *Y4*, *Y5*, *Y6*, *Y7*, *Y8*

Graphs columns (vectors) of array parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParX

Name of the array parameter whose column vector values will be the abscissa of the graph. If blank, row subscript numbers are used instead. *ParX* is not sorted by the program.

ParY

Name of the array parameter whose column vector values will be graphed against the *ParX* values.

Y2, *Y3*, *Y4*, *Y5*, *Y6*, *Y7*, *Y8*
Additional column subscript of the *ParY* array parameter whose values are to be graphed against the *ParX* values.

Notes

The column to be graphed and the starting row for each array parameter must be specified as subscripts. Additional columns of the *ParY* array parameter may be graphed by specifying column numbers for *Y2*, *Y3*, ...,*Y8*. For example, ***VPLOT**,TIME (4,6), DISP (8,1),2,3 specifies that the 1st, 2nd, and 3rd columns of array parameter DISP (all starting at row 8) are to be graphed against the 6th column of array parameter TIME (starting at row 4). The columns will be graphed from the starting row to their maximum extent. See the ***VLEN** and ***VMASK** commands to limit or skip data to be graphed. The array parameters specified on the ***VPLOT** command must be of the same type (type ARRAY or TABLE; [***DIM**]. Arrays of type TABLE will be graphed as continuous curves. Arrays of type ARRAY will be displayed in bar chart fashion.

The normal curve labeling scheme for ***VPLOT** is to label curve 1 "COL 1", curve 2 "COL 2" and so on. You can use the **/GCOLUMN** command to apply user-specifed labels (8 characters maximum) to your curves. See Modifying Curve Labels in the *ANSYS APDL Programmer's Guide* for more information on using **/GCOLUMN**.

When a graph plot reaches minimum or maximum y-axis limits, ANSYS indicates the condition by clipping the graph. The clip appears as a horizontal magenta line. ANSYS calculates y-axis limits automatically; however, you can modify the (YMIN and YMAX) limits via the **/YRANGE** command.

This command is valid in any processor.

Menu Paths

Utility Menu>Plot>Array Parameters

***VPUT**, *ParR*, *Entity*, *ENTNUM*, *Item1*, *IT1NUM*, *Item2*, *IT2NUM*, *KLOOP*

Restores array parameter values into the ANSYS database.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR
The name of the input vector array parameter. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**] with data input.

Entity
Entity keyword. Valid keywords are shown for *Entity* = in the table below.

ENTNUM
The number of the entity (as shown for *ENTNUM*= in the table below).

Item1
The name of a particular item for the given entity. Valid items are as shown in the *Item1* columns of the table below.

IT1NUM
The number (or label) for the specified *Item1* (if any). Valid *IT1NUM* values are as shown in the *IT1NUM* columns of the table below. Some *Item1* labels do not require an *IT1NUM* value.

Item2, *IT2NUM*
A second set of item labels and numbers to further qualify the item for which data is to be stored. Most items do not require this level of information.

KLOOP
Field to be looped on:

0 or 2 --
Loop on the *ENTNUM* field (default).

3 --
Loop on the *Item1* field.

4 --
Loop on the *IT1NUM* field. Successive items are as shown with *IT1NUM*.

5 --
Loop on the *Item2* field.

6 --
Loop on the *IT2NUM* field. Successive items are as shown with *IT2NUM*.

Notes

The ***VPUT** command is not supported for PowerGraphics displays. Inconsistent results may be obtained if this command is not used in **/GRAPHICS**, FULL.

Plot and print operations entered via the GUI (**Utility Menu> Pltcrtls, Utility Menu> Plot**) incorporate the **AVPRIN** command. This means that the principal and equivalent values are recalculated. If you use ***VPUT** to put data back into the database, issue the plot commands from the command line to preserve your data.

This operation is basically the inverse of the ***VGET** operation. Vector items are put directly (without any coordinate system transformation) into the ANSYS database. Items can only replace existing items of the database and not create new items. Degree of freedom results that are replaced in the database are available for all subsequent postprocessing operations. Other results are changed temporarily and are available mainly for the immediately following print and display operations. The vector specification ***VCUM** does not apply to this command. The valid labels for the location fields (*Entity*, *ENTNUM*, *Item1*, and *IT1NUM*) are listed below. *Item2* and *IT2NUM* are not currently used. Not all items from the ***VGET** list are allowed on ***VPUT** since putting values into some locations could cause the database to be inconsistent.

This command is valid in any processor.

*VPUT - POST1 Items

***Entity* = NODE, *ENTNUM* = *n* (node number)**

Item1	IT1NUM	Description
Valid labels for nodal degree of freedom results are:		
U	X, Y, Z	X, Y, or Z structural displacement.
ROT	X, Y, Z	X, Y, or Z structural rotation.
TEMP		Temperature. For SHELL131 and SHELL132 elements with KEYOPT(3) = 0 or 1, use TBOT, TE2, TE3, . . ., TTOP instead of TEMP. Alternative get functions: TEMP(N), TBOT(N), TE2(N), etc.
PRES		Pressure.

Entity **= NODE,** ***ENTNUM*** **=** ***n*** **(node number)**

Item1	IT1NUM	Description
VOLT		Electric potential.
MAG		Magnetic scalar potential.
V	X, Y, Z	X, Y, or Z fluid velocity.
A	X, Y, Z	X, Y, or Z magnetic vector potential.
CURR		Current.
EMF		Electromotive force drop.
ENKE		Turbulent kinetic energy (FLOTRAN).
ENDS		Turbulent energy dissipation (FLOTRAN).

Valid labels for element nodal results are:

Item1	IT1NUM	Description
S	X, Y, Z, XY, YZ, XZ	Component stress.
"	1, 2, 3	Principal stress.
"	INT, EQV	Stress intensity or equivalent stress.
EPTO	X, Y, Z, XY, YZ, XZ	Component total strain (EPEL + EPPL + EPCR).
"	1,2,3	Principal total strain.
"	INT, EQV	Total strain intensity or total equivalent strain.
EPEL	X, Y, Z, XY, YZ, XZ	Component elastic strain.
"	1, 2, 3	Principal elastic strain.
"	INT, EQV	Elastic strain intensity or elastic equivalent strain.
EPPL	X, Y, Z, XY, YZ, XZ	Component plastic strain.
"	1,2,3	Principal plastic strain.
"	INT, EQV	Plastic strain intensity or plastic equivalent strain.
EPCR	X, Y, Z, XY, YZ, XZ	Component creep strain.
"	1, 2, 3	Principal creep strain.
"	INT, EQV	Creep strain intensity or creep equivalent strain.
EPTH	X, Y, Z, XY, YZ, XZ	Component thermal strain.
"	1, 2, 3	Principal thermal strain.
"	INT, EQV	Thermal strain intensity or thermal equivalent strain.
EPSW		Swelling strain.
NL	SEPL	Equivalent stress (from stress-strain curve).
"	SRAT	Stress state ratio.
"	HPRES	Hydrostatic pressure.
"	EPEQ	Accumulated equivalent plastic strain.
"	PSV	Plastic state variable.
"	PLWK	Plastic work/volume.
TG	X, Y, Z	Component thermal gradient.
TF	X, Y, Z	Component thermal flux.
PG	X, Y, Z	Component pressure gradient.
EF	X, Y, Z	Component electric field.
D	X, Y, Z	Component electric flux density.

Entity **= NODE, *ENTNUM* = *n* (node number)**

Item1	IT1NUM	Description
H	X, Y, Z	Component magnetic field intensity.
B	X, Y, Z	Component magnetic flux density.
FMAG	X, Y, Z	Component magnetic force.

Valid labels for FLOTRAN nodal results are:

Item1	IT1NUM	Description
TTOT		Total temperature.
HFLU		Heat flux.
HFLM		Heat transfer (film) coefficient.
COND		Fluid laminar conductivity.
PCOE		Pressure coefficient.
PTOT		Total (stagnation) pressure.
MACH		Mach number.
STRM		Stream function. (2-D applications only.)
DENS		Fluid density.
VISC		Fluid laminar viscosity.
EVIS		Fluid effective viscosity.
ECON		Fluid effective conductivity.
YPLU		Y+, a turbulent law of the wall parameter.
TAUW		Shear stress at the wall.

Entity = ELEM, *ENTNUM* = *n* (element number)

Valid labels for element results are:

ETAB	Lab	Any user-defined element table label (see **ETABLE** command).

Menu Paths

Utility Menu>Parameters>Array Operations>Put Array Data

***VREAD**, *ParR*, *Fname*, *Ext*, *--*, *Label*, *n1*, *n2*, *n3*, *NSKIP*

Reads data and produces an array parameter vector or matrix.

APDL: Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR

The name of the resulting array parameter vector. See ***SET** for name restrictions. The parameter must exist as a dimensioned array [***DIM**]. String arrays are limited to a maximum of 8 characters.

Fname

File name and directory path (248 characters maximum, including directory). If you do not specify a directory path, it will default to your working directory and you can use all 248 characters for the file name.

If the *Fname* field is left blank, reading continues from the current input device, such as the terminal.

Ext
Filename extension (8 character maximum).

--
Unused field

Label
Can take a value of IJK, IKJ, JIK, JKI, KIJ, KJI, or blank (IJK).

n1, *n2*, *n3*
Read as (((*ParR* (i,j,k), k = 1,n1), i = 1, n2), j = 1, n3) for *Label* = KIJ. *n2* and *n3* default to 1.

NSKIP
Number of lines at the beginning of the file being read that will be skipped during the reading. Default = 0.

Notes

Reads data from a file and fills in an array parameter vector or matrix. Data are read from a formatted file or, if the menu is off [**/MENU**,OFF] and *Fname* is blank, from the next input lines. The format of the data to be read must be input immediately following the ***VREAD** command. The format specifies the number of fields to be read per record, the field width, and the placement of the decimal point (if none specified in the value). The read operation follows the available FORTRAN FORMAT conventions of the system (see your system FORTRAN manual). Any standard FORTRAN *real* format (such as (4F6.0), (E10.3,2X,D8.2), etc.) or alphanumeric format (A) may be used. Alphanumeric strings are limited to a maximum of 8 characters for any field (A8). Integer (I) and list-directed (*) descriptors may *not* be used. *The parentheses must be included in the format* and the format must not exceed 80 characters (including parentheses). The input line length is limited to 128 characters.

A starting array element number must be defined for the result array parameter vector (numeric or character). For example, entering these two lines:

```
*VREAD,A(1),ARRAYVAL
(2F6.0)
```

will read two values from each line of file ARRAYVAL and assign the values to A(1), A(2), A(3), etc. Reading continues until successive row elements [***VLEN**, ***VMASK**, ***DIM**] are filled.

For an array parameter matrix, a starting array element row and column number must be defined. For example, entering these two lines:

```
VREAD,A(1,1),ARRAYVAL,,,IJK,10,2
(2F6.0)
```

will read two values from each line of file ARRAYVAL and assign the values to A(1,1), A(2,1), A(3,1), etc. Reading continues until *n1* (10) successive row elements are filled. Once the maximum row number is reached, subsequent data will be read into the next column (e.g., A(1,2), A(2,2), A(3,2), etc.)

For numerical parameters, absolute values and scale factors may be applied to the result parameter [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. See the ***VOPER** command for details. If you are in the GUI the ***VREAD** command must be contained in an externally prepared file read into the ANSYS program (i.e., ***USE**, **/INPUT**, etc.).

This command is not applicable to 4- or 5-D arrays.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Parameters>Read from File

---

*VSCFUN, *ParR*, *Func*, *Par1*

Determines properties of an array parameter.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

ParR

The name of the resulting scalar parameter. See ***SET** for name restrictions.

Func

Functions:

MAX --

Maximum: the maximum *Par1* array element value.

MIN --

Minimum: the minimum *Par1* array element value.

LMAX --

Index location of the maximum *Par1* array element value. Array *Par1* is searched starting from its specified index.

LMIN --

Index location of the minimum *Par1* array element value. Array *Par1* is searched starting from its specified index.

FIRST --

Index location of the first nonzero value in array *Par1*. Array *Par1* is searched starting from its specified index.

LAST --

Index location of the last nonzero value in array *Par1*. Array *Par1* is searched starting from its specified index.

SUM --

Sum: *Par1* (the summation of the *Par1* array element values).

MEDI --

Median: value of *Par1* at which there are an equal number of values above and below.

MEAN --

Mean: (σ Par1)/NUM, where NUM is the number of summed values.

VARI --

Variance: (σ ((*Par1*-MEAN)**2))/NUM.

STDV --

Standard deviation: square root of VARI.

RMS --

Root-mean-square: square root of (σ (*Par1***2))/NUM.

NUM --

Number: the number of summed values (masked values are not counted).

Par1
Array parameter vector in the operation.

Notes

Operates on one input array parameter vector and produces one output scalar parameter according to:

ParR = f(*Par1*)

where the functions (f) are described below. The starting array element number must be defined for the array parameter vector. For example, ***VSCFUN**,MU,MEAN,A(1) finds the mean of the A vector values, starting from the first value and stores the result as parameter MU. Operations use successive array elements [***VLEN**, ***VMASK**] with the default being all successive array elements. Absolute values and scale factors may be applied to all parameters [***VABS**, ***VFACT**]. Results may be cumulative [***VCUM**]. See the ***VOPER** command for details.

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Operations>Vector-Scalar Func

---

*VSTAT
Lists the current specifications for the array parameters.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

Notes

Lists the current specifications for the ***VABS**, ***VCOL**, ***VCUM**, ***VFACT**, ***VLEN**, and ***VMASK** commands.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

---

*VWRITE, *Par1*, *Par2*, *Par3*, *Par4*, *Par5*, *Par6*, *Par7*, *Par8*, *Par9*, *Par10*, *Par11*, *Par12*, *Par13*, *Par14*, *Par15*, *Par16*, *Par17*, *Par18*, *Par19*
Writes data to a file in a formatted sequence.

APDL: Array Parameters

MP ME ST DY <> PR EM <> FL PP ED

Par1 - Par19
You can write up to 19 parameters (or constants) at a time. Any *Par* values after a blank *Par* value are ignored. If you leave them all blank, one line will be written (to write a title or a blank line). If you input the keyword SEQU, a sequence of numbers (starting from 1) will be written for that item.

Notes

You use ***VWRITE** to write data to a file in a formatted sequence. Data items (*Par1*, *Par2*, etc.) may be array parameters, scalar parameters, character parameters (scalar or array), or constants. You must evaluate expressions and functions in the data item fields before using the ***VWRITE** command, since initially they will be evaluated to a constant and remain constant throughout the operation. Unless a file is defined with the ***CFOPEN** command, data is written to the standard output file. Data written to the standard output file may be diverted to a different file by first switching the current output file with the **/OUTPUT** command. You can also use the ***MWRITE** command to write data to a specified file. Both commands contain format descriptors on the line immediately following the command. The format descriptors can be in either Fortran or C format.

You must enclose Fortran format descriptors in parentheses. They must immediately follow the ***VWRITE** command on a separate line of the same input file. Do not include the word FORMAT. The format must specify the number of fields to be written per line, the field width, the placement of the decimal point, etc. You should use one field descriptor for each data item written. The write operation uses your system's available FORTRAN FORMAT conventions (see your system FORTRAN manual). You can use any standard FORTRAN *real* format (such as (4F6.0), (E10.3,2X,D8.2), etc.) and alphanumeric format (A). Alphanumeric strings are limited to a maximum of 8 characters for any field (A8) using the Fortran format. Use the "C" format for string arrays larger than 8 characters. Integer (I) and list-directed (*) descriptors may *not* be used. You can include text in the format as a quoted string. *The parentheses must be included in the format* and the format must not exceed 80 characters (including parentheses). The output line length is limited to 128 characters.

The "C" format descriptors are used if the first line of the format descriptor is not a left parenthesis. "C" format descriptors are up to 80 characters long, consisting of text strings and predefined "data descriptors" between the strings where numeric or alphanumeric character data will be inserted. The normal descriptors are %I for integer data, %G for double precision data, %C for alphanumeric character data, and %/ for a line break. Each descriptor must be preceded by a blank. There must be one data descriptor for each specified value (8 maximum) in the order of the specified values. The enhanced formats described in ***MSG** may also be used.

For array parameter items, you must define the starting array element number. Looping continues (incrementing the vector index number of each array parameter by one) each time you output a line, until the maximum array vector element is written. For example, ***VWRITE**,A(1) followed by (F6.0) will write one value per output line, i.e., A(1), A(2), A(3), A(4), etc. You write constants and scalar parameters with the same values for each loop. You can also control the number of loops and loop skipping with the ***VLEN** and ***VMASK** commands. The vector specifications ***VABS**, ***VFACT**, and ***VCUM** do not apply to this command. If looping continues beyond the supplied data array's length, zeros will be output for numeric array parameters and blanks for character array parameters. For multi-dimensioned array parameters, only the first (row) subscript is incremented. See the ***VOPER** command for details. If you are in the GUI, the ***VWRITE** command must be contained in an externally prepared file and read into ANSYS (i.e., ***USE**, **/INPUT**, etc.).

This command is valid in any processor.

Menu Paths

Utility Menu>Parameters>Array Parameters>Write to File

/WAIT, *DTIME*

Causes a delay before the reading of the next command.

APDL: Process Controls

MP ME ST DY <> PR EM <> FL PP ED

DTIME

Time delay (in seconds). Maximum time delay is 59 seconds.

Notes

The command following the **/WAIT** will not be processed until the specified wait time increment has elapsed. Useful when reading from a prepared input file to cause a pause, for example, after a display command so that the display can be reviewed for a period of time. Another "wait" feature is available via the ***ASK** command.

This command is valid in any processor.

Menu Paths

This command cannot be accessed from a menu.

Appendix A. APDL Gateway Commands

When you need to determine the applicability of a command or a group of commands to a specific product, the following *GET functions will return a TRUE or a FALSE (a 1 or a 0) value to indicate if the command in question is valid for your ANSYS product.

```
  Additional *get commands for a new entity=PRODUCT

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Item2    It2num  Description
pname                                 -P option from Ansys command line
name                 start    1-n     Ansys product name.  A character string
                                      of 8 characters is returned starting at
                                      position It2num. Use *dim and *do to get
                                      all 32 characters.
Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
/aux12               Check for Ansys gateway command/feature /AUX12
/config              Check for Ansys gateway command/feature /CONFIG
/ucmd                Check for Ansys gateway command/feature /UCMD
addam                Check for Ansys gateway command/feature ADDAM
alphad               Check for Ansys gateway command/feature ALPHAD
antype               Check for Ansys gateway command/feature ANTYPE
antype   static      Check for Ansys gateway command/feature ANTYPE,STATIC
antype   buckle      Check for Ansys gateway command/feature ANTYPE,BUCKLE
antype   modal       Check for Ansys gateway command/feature ANTYPE,MODAL
antype   harmic      Check for Ansys gateway command/feature ANTYPE,HARMIC
antype   trans       Check for Ansys gateway command/feature ANTYPE,TRANS
antype   substr      Check for Ansys gateway command/feature ANTYPE,SUBSTR
antype   spectr      Check for Ansys gateway command/feature ANTYPE,SPECTR
arclen               Check for Ansys gateway command/feature ARCLEN
betad                Check for Ansys gateway command/feature BETAD
blc4                 Check for Ansys gateway command/feature BLC4
blc5                 Check for Ansys gateway command/feature BLC5
block                Check for Ansys gateway command/feature BLOCK
cdread               Check for Ansys gateway command/feature CDREAD
con4                 Check for Ansys gateway command/feature CON4
cone                 Check for Ansys gateway command/feature CONE
cqc                  Check for Ansys gateway command/feature CQC
cyl4                 Check for Ansys gateway command/feature CYL4
cyl5                 Check for Ansys gateway command/feature CYL5
cylind               Check for Ansys gateway command/feature CYLIND


Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
damorph              Check for Ansys gateway command/feature DAMORPH
demorph              Check for Ansys gateway command/feature DEMORPH
dsum                 Check for Ansys gateway command/feature DSUM
dvmorph              Check for Ansys gateway command/feature DVMORPH
edadapt              Check for Ansys gateway command/feature EDADAPT
edbvis               Check for Ansys gateway commands/feature EDBVIS
eddc                 Check for Ansys gateway commands/feature EDDC
edcgen               Check for Ansys gateway commands/feature EDCGEN
edclist              Check for Ansys gateway commands/feature EDCLIST
edcontact            Check for Ansys gateway commands/feature EDCONTACT
edcpu                Check for Ansys gateway commands/feature EDCPU
edcrb                Check for Ansys gateway commands/feature EDCRB
edcsc                Check for Ansys gateway commands/feature EDCSC
edcts                Check for Ansys gateway commands/feature EDCTS
edcurve              Check for Ansys gateway commands/feature EDCURVE
eddamp               Check for Ansys gateway commands/feature EDDAMP
edenergy             Check for Ansys gateway commands/feature EDENERGY
edfplot              Check for Ansys gateway commands/feature EDFPLOT
edhgls               Check for Ansys gateway commands/feature EDHGLS
edhtime              Check for Ansys gateway commands/feature EDHTIME
```

```
edhist              Check for Ansys gateway commands/feature EDHIST
edint               Check for Ansys gateway commands/feature EDINT
edvel               Check for Ansys gateway commands/feature EDVEL
edlcs               Check for Ansys gateway commands/feature EDLCS
edldplot            Check for Ansys gateway commands/feature EDLDPLOT

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num     Description (return values:  1=allowed,  0=not allowed)
edload              Check for Ansys gateway commands/feature EDLOAD
edmp                Check for Ansys gateway commands/feature EDMP
ednb                Check for Ansys gateway commands/feature EDNB
edndtsd             Check for Ansys gateway commands/feature EDNDTSD
edout               Check for Ansys gateway commands/feature EDOUT
edpart              Check for Ansys gateway commands/feature EDPART
edread              Check for Ansys gateway commands/feature EDREAD
eddrelax            Check for Ansys gateway commands/feature EDDRELAX
edrst               Check for Ansys gateway commands/feature EDRST
edshell             Check for Ansys gateway commands/feature EDSHELL
edsolve             Check for Ansys gateway commands/feature EDSOLVE
edstart             Check for Ansys gateway commands/feature EDSTART
edweld              Check for Ansys gateway commands/feature EDWELD
edwrite             Check for Ansys gateway commands/feature EDWRITE
ekill               Check for Ansys gateway commands/feature EKILL
emis                Check for Ansys gateway commands/feature EMIS
et                  Check for Ansys gateway commands/feature ET



Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num     Description (return values:  1=allowed,  0=not allowed)
etchg               Check for Ansys gateway commands/feature ETCHG
fldata              Check for Ansys gateway commands/feature FLDATA
flotest             Check for Ansys gateway commands/feature FLOTEST
flread              Check for Ansys gateway commands/feature FLREAD
fvmesh              Check for Ansys gateway commands/feature FVMESH
grp                 Check for Ansys gateway commands/feature GRP
hropt               Check for Ansys gateway commands/feature HROPT
hropt    full       Check for Ansys gateway commands/feature HROPT,FULL
hropt    reduc      Check for Ansys gateway commands/feature HROPT,REDUC
hropt    msup       Check for Ansys gateway commands/feature HROPT,MSUP
igesin              Check for Ansys gateway commands/feature IGESIN
igesout             Check for Ansys gateway commands/feature IGESOUT
modopt              Check for Ansys gateway commands/feature MODOPT
modopt   reduc      Check for Ansys gateway commands/feature MODOPT,REDUC
modopt   subsp      Check for Ansys gateway commands/feature MODOPT,SUBSP
modopt   unsym      Check for Ansys gateway commands/feature MODOPT,UNSYM
modopt   damp       Check for Ansys gateway commands/feature MODOPT,DAMP
modopt   lanb       Check for Ansys gateway commands/feature MODOPT,LANB
modopt   qrdamp     Check for Ansys gateway commands/feature MODOPT,QRDAMP
mooney              Check for Ansys gateway commands/feature MOONEY
mp                  Check for Ansys gateway commands/feature MP
mp       ex         Check for Ansys gateway commands/feature MP,EX
mp       alpx       Check for Ansys gateway commands/feature MP,ALPX
mp       reft       Check for Ansys gateway commands/feature MP,REFT
mp       prxy       Check for Ansys gateway commands/feature MP,PRXY

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num     Description (return values:  1=allowed,  0=not allowed)
mp       nuxy       Check for Ansys gateway commands/feature MP,NUXY
mp       gxy        Check for Ansys gateway commands/feature MP,GXY
mp       damp       Check for Ansys gateway commands/feature MP,DAMP
mp       mu         Check for Ansys gateway commands/feature MP,MU
mp       dens       Check for Ansys gateway commands/feature MP,DENS
mp       c          Check for Ansys gateway commands/feature MP,C
mp       enth       Check for Ansys gateway commands/feature MP,ENTH
mp       kxx        Check for Ansys gateway commands/feature MP,KXX
mp       hf         Check for Ansys gateway commands/feature MP,HF
mp       emis       Check for Ansys gateway commands/feature MP,EMIS
mp       qrate      Check for Ansys gateway commands/feature MP,QRATE
mp       visc       Check for Ansys gateway commands/feature MP,VISC
mp       sonc       Check for Ansys gateway commands/feature MP,SONC
mp       rsvx       Check for Ansys gateway commands/feature MP,RSVX
```

```
mp       perx        Check for Ansys gateway commands/feature MP,PERX
mp       murx        Check for Ansys gateway commands/feature MP,MURX
mp       mgxx        Check for Ansys gateway commands/feature MP,MGXX
mp       hgls        Check for Ansys gateway commands/feature MP,HGLS
mp       rigid       Check for Ansys gateway commands/feature MP,RIGID
mp       cable       Check for Ansys gateway commands/feature MP,CABLE
mp       ortho       Check for Ansys gateway commands/feature MP,ORTHO
mp       lsst        Check for Ansys gateway commands/feature MP,LSST
mpdata               Check for Ansys gateway commands/feature MPDATA
mpdata   ex          Check for Ansys gateway commands/feature MPDATA,EX
mpdata   alpx        Check for Ansys gateway commands/feature MPDATA,ALPX

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
mpdata   reft        Check for Ansys gateway commands/feature MPDATA,REFT
mpdata   prxy        Check for Ansys gateway commands/feature MPDATA,PRXY
mpdata   nuxy        Check for Ansys gateway commands/feature MPDATA,NUXY
mpdata   gxy         Check for Ansys gateway commands/feature MPDATA,GXY
mpdata   damp        Check for Ansys gateway commands/feature MPDATA,DAMP
mpdata   mu          Check for Ansys gateway commands/feature MPDATA,MU
mpdata   dens        Check for Ansys gateway commands/feature MPDATA,DENS
mpdata   c           Check for Ansys gateway commands/feature MPDATA,C
mpdata   enth        Check for Ansys gateway commands/feature MPDATA,ENTH
mpdata   kxx         Check for Ansys gateway commands/feature MPDATA,KXX
mpdata   hf          Check for Ansys gateway commands/feature MPDATA,HF
mpdata   emis        Check for Ansys gateway commands/feature MPDATA,EMIS
mpdata   qrate       Check for Ansys gateway commands/feature MPDATA,QRATE
mpdata   visc        Check for Ansys gateway commands/feature MPDATA,VISC
mpdata   sonc        Check for Ansys gateway commands/feature MPDATA,SONC
mpdata   rsvx        Check for Ansys gateway commands/feature MPDATA,RSVX
mpdata   perx        Check for Ansys gateway commands/feature MPDATA,PERX
mpdata   murx        Check for Ansys gateway commands/feature MPDATA,MURX
mpdata   mgxx        Check for Ansys gateway commands/feature MPDATA,MGXX
mpdata   lsst        Check for Ansys gateway commands/feature MPDATA,LSST
mscap                Check for Ansys gateway commands/feature MSCAP
msdata               Check for Ansys gateway commands/feature MSDATA
msmeth               Check for Ansys gateway commands/feature MSMETH
msnomf               Check for Ansys gateway commands/feature MSNOMF
msprop               Check for Ansys gateway commands/feature MSPROP

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
msquad               Check for Ansys gateway commands/feature MSQUAD
msrelax              Check for Ansys gateway commands/feature MSRELAX
mssolu               Check for Ansys gateway commands/feature MSSOLU
msspec               Check for Ansys gateway commands/feature MSSPEC
msvary               Check for Ansys gateway commands/feature MSVARY
nlgeom               Check for Ansys gateway commands/feature NLGEOM
nrlsum               Check for Ansys gateway commands/feature NRLSUM
optyp                Check for Ansys gateway commands/feature OPTYP
optyp    subp        Check for Ansys gateway commands/feature OPTYP,SUBP
optyp    first       Check for Ansys gateway commands/feature OPTYP,FIRST
optyp    rand        Check for Ansys gateway commands/feature OPTYP,RAND
optyp    run         Check for Ansys gateway commands/feature OPTYP,RUN
optyp    fact        Check for Ansys gateway commands/feature OPTYP,FACT
optyp    grad        Check for Ansys gateway commands/feature OPTYP,GRAD
optyp    sweep       Check for Ansys gateway commands/feature OPTYP,SWEEP
optyp    user        Check for Ansys gateway commands/feature OPTYP,USER
opuser               Check for Ansys gateway commands/feature OPUSER
pri2                 Check for Ansys gateway commands/feature PRI2
prism                Check for Ansys gateway commands/feature PRISM
psdcom               Check for Ansys gateway commands/feature PSDCOM
psdfrq               Check for Ansys gateway commands/feature PSDFRQ
psolve               Check for Ansys gateway commands/feature PSOLVE
rate                 Check for Ansys gateway commands/feature RATE
resume               Check for Ansys gateway commands/feature RESUME
rpr4                 Check for Ansys gateway commands/feature RPR4

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
rprism               Check for Ansys gateway commands/feature RPRISM
save                 Check for Ansys gateway commands/feature SAVE
```

```
se                  Check for Ansys gateway commands/feature SE
sesymm              Check for Ansys gateway commands/feature SESYMN
setran              Check for Ansys gateway commands/feature SETRAN
solve               Check for Ansys gateway commands/feature SOLVE
sph4                Check for Ansys gateway commands/feature SPH4
sph5                Check for Ansys gateway commands/feature SPH5
sphere              Check for Ansys gateway commands/feature SPHERE
spop                Check for Ansys gateway commands/feature SPOP
spop     sprs       Check for Ansys gateway commands/feature SPOP,SPRS
spop     mprs       Check for Ansys gateway commands/feature SPOP,MPRS
spop     ddam       Check for Ansys gateway commands/feature SPOP,DDAM
spop     psd        Check for Ansys gateway commands/feature SPOP,PSD
srss                Check for Ansys gateway commands/feature SRSS
tb                  Check for Ansys gateway commands/feature TB
tb       bkin       Check for Ansys gateway commands/feature TB,BKIN
tb       mkin       Check for Ansys gateway commands/feature TB,MKIN
tb       miso       Check for Ansys gateway commands/feature TB,MISO
tb       biso       Check for Ansys gateway commands/feature TB,BISO
tb       aniso      Check for Ansys gateway commands/feature TB,ANISO
tb       dp         Check for Ansys gateway commands/feature TB,DP
tb       anand      Check for Ansys gateway commands/feature TB,ANAND
tb       melas      Check for Ansys gateway commands/feature TB,MELAS
tb       user       Check for Ansys gateway commands/feature TB,USER

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num     Description (return values:  1=allowed,  0=not allowed)
tb       creep      Check for Ansys gateway commands/feature TB,CREEP
tb       swell      Check for Ansys gateway commands/feature TB,SWELL
tb       bh         Check for Ansys gateway commands/feature TB,BH
tb       piez       Check for Ansys gateway commands/feature TB,PIEZ
tb       fail       Check for Ansys gateway commands/feature TB,FAIL
tb       mooney     Check for Ansys gateway commands/feature TB,MOONEY
tb       water      Check for Ansys gateway commands/feature TB,WATER
tb       anel       Check for Ansys gateway commands/feature TB,ANEL
tb       concr      Check for Ansys gateway commands/feature TB,CONCR
tb       pflow      Check for Ansys gateway commands/feature TB,PFLOW
tb       evisc      Check for Ansys gateway commands/feature TB,EVISC
tb       plaw       Check for Ansys gateway commands/feature TB,PLAW
tb       foam       Check for Ansys gateway commands/feature TB,FOAM
tb       honey      Check for Ansys gateway commands/feature TB,HONEY
tb       comp       Check for Ansys gateway commands/feature TB,COMP
tb       nl         Check for Ansys gateway commands/feature TB,NL
tb       nliso      Check for Ansys gateway commands/feature TB,NLISO
tb       chab       Check for Ansys gateway commands/feature TB,CHAB
tb       boyce      Check for Ansys gateway commands/feature TB,BOYCE
tb       eos        Check for Ansys gateway commands/feature TB,EOS
torus               Check for Ansys gateway commands/feature TORUS
trnopt              Check for Ansys gateway commands/feature TRNOPT
trnopt   full       Check for Ansys gateway commands/feature TRNOPT,FULL
trnopt   reduc      Check for Ansys gateway commands/feature TRNOPT,REDUC
trnopt   msup       Check for Ansys gateway commands/feature TRNOPT,MSUP

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num     Description (return values:  1=allowed,  0=not allowed)
usrcal              Check for Ansys gateway commands/feature USRCAL
v                   Check for Ansys gateway commands/feature V
va                  Check for Ansys gateway commands/feature VA
vadd                Check for Ansys gateway commands/feature VADD
vcvfill             Check for Ansys gateway commands/feature VCVFILL
vdrag               Check for Ansys gateway commands/feature VDRAG
vext                Check for Ansys gateway commands/feature VEXT
vgen                Check for Ansys gateway commands/feature VGEN
vglue               Check for Ansys gateway commands/feature VGLUE
vinp                Check for Ansys gateway commands/feature VINP
vinv                Check for Ansys gateway commands/feature VINV
vlscale             Check for Ansys gateway commands/feature VLSCALE
vmesh               Check for Ansys gateway commands/feature VMESH
voffset             Check for Ansys gateway commands/feature VOFFSET
vovlap              Check for Ansys gateway commands/feature VOVLAP
vptn                Check for Ansys gateway commands/feature VPTN
vrotat              Check for Ansys gateway commands/feature VROTAT
vsba                Check for Ansys gateway commands/feature VSBA
```

```
vsbv                Check for Ansys gateway commands/feature VSBV
vsbw                Check for Ansys gateway commands/feature VSBW
vsymm               Check for Ansys gateway commands/feature VSYMM
vtran               Check for Ansys gateway commands/feature VTRAN

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
elem     i           Check to see if Ansys element type "i" is allowed.

Entity=PRODUCT,ENTNUM=0 (or blank)
Item1    It1num      Description (return values:  1=allowed,  0=not allowed)
limit  node   Get maximum allowed node number
limit  elem   Get maximum allowed element number
limit  kp      Get maximum allowed keypoint number
limit  line   Get maximum allowed line number
limit  area   Get maximum allowed area number
limit  vol   Get maximum allowed volume number
limit  dof   Get maximum allowed dof number
limit  mdof   Get maximum allowed master dof number
```

Appendix B. GET Function Summary

A "get function" is available for some items, and can be used instead of the ***GET** command. The function returns the value and uses it where the function is input (bypassing the need for storing the value with a parameter name and inputting the parameter name where the value is to be used). For example, assume the average X-location of two nodes is to be calculated. Using the ***GET** command, parameter L1 can be assigned the X location of node 1 (***GET**, L1, NODE, 1, LOC, X), and parameter L2 can be assigned the X location of node 2, then the mid location can be computed from MID = (L1+L2)/2. However, using the node location "get function" NX(N), which returns the X location of node N, MID can be computed directly from MID = (NX(1)+NX(2))/2, without the need for intermediate parameters L1 and L2. Get functions return values in the active coordinate system unless stated otherwise.

Get function arguments may themselves be parameters or other get functions. The get function NELEM(ENUM,NPOS) returns the node number in position NPOS for element ENUM. Combining functions, NX(NELEM(ENUM,NPOS)) returns the X location of that node. Get functions (where available) are shown with the corresponding ***GET** items in the tables below and are summarized at the end of this command description.

Get functions are described at the beginning of this command (see Notes) and are shown as alternatives to the ***GET** items where they apply. They are summarized here (grouped by functionality) for convenience.

Table B.1 *GET - Get Function Summary

"Get Function" Summary	
Entity Status Get Function	**Description**
NSEL(*N*)	Status of node *N*: -1=unselected, 0=undefined, 1=selected.
ESEL(*E*)	Status of element *E*: -1=unselected, 0=undefined, 1=selected.
KSEL(*K*)	Status of keypoint *K*: -1=unselected, 0=undefined, 1=selected.
LSEL(*L*)	Status of line *L*: -1=unselected, 0=undefined, 1=selected.
ASEL(*A*)	Status of area *A*: -1=unselected, 0=undefined, 1=selected.
VSEL(*V*)	Status of volume *V*: -1=unselected, 0=undefined, 1=selected.
Next Selected Entity	
NDNEXT(*N*)	Next selected node having a node number greater than *N*.
ELNEXT(*E*)	Next selected element having an element number greater than *E*.
KPNEXT(*K*)	Next selected keypoint having a keypoint number greater than *K*.
LSNEXT(*L*)	Next selected line having a line number greater than *L*.
ARNEXT(*A*)	Next selected area having an area number greater than *A*.
VLNEXT(*V*)	Next selected volume having a volume number greater than *V*.
Locations	
CENTRX(*E*)	Centroid X-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
CENTRY(*E*)	Centroid Y-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
CENTRZ(*E*)	Centroid Z-coordinate of element *E* in global Cartesian coordinate system. Centroid is determined from the selected nodes on the element.
NX(*N*)	X-coordinate of node *N* in the active coordinate system.
NY(*N*)	Y-coordinate of node *N* in the active coordinate system.
NZ(*N*)	Z-coordinate of node *N* in the active coordinate system.

"Get Function" Summary

Entity Status Get Function	Description
KX(*K*)	X-coordinate of keypoint *K* in the active coordinate system
KY(*K*)	Y-coordinate of keypoint *K* in the active coordinate system
KZ(*K*)	Z-coordinate of keypoint *K* in the active coordinate system
LX(*L*, *LFRAC*)	X-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0).
LY(*L*, *LFRAC*)	Y-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0).
LZ(*L*, *LFRAC*)	Z-coordinate of line *L* at length fraction *LFRAC* (0.0 to 1.0).
LSX(*L*, *LFRAC*)	X slope of line *L* at length fraction *LFRAC* (0.0 to 1.0).
LSY(*L*, *LFRAC*)	Y slope of line *L* at length fraction *LFRAC* (0.0 to 1.0).
LSZ(*L*, *LFRAC*)	Z slope of line *L* at length fraction *LFRAC* (0.0 to 1.0).
Nearest to Location	
NODE(*X*, *Y*, *Z*)	Number of the selected node nearest the *X*, *Y*, *Z* point (in the active coordinate system, lowest number for coincident nodes).
KP(*X*, *Y*, *Z*)	Number of the selected keypoint nearest the *X*, *Y*, *Z* point (in the active coordinate system, lowest number for coincident nodes).
Distances	
DISTND(*N1*, *N2*)	Distance between nodes *N1* and *N2*.
DISTKP(*K1*, *K2*)	Distance between keypoints *K1* and *K2*.
DISTEN(*E*, *N*)	Distance between the centroid of element *E* and node *N*. Centroid is determined from the selected nodes on the element.
Angles (in radians by default -- see the *AFUN command)	
ANGLEN(*N1*, *N2*, *N3*)	Subtended angle between two lines (defined by three nodes where *N1* is the vertex node). Default is in radians.
ANGLEK(*K1*, *K2*, *K3*)	Subtended angle between two lines (defined by three keypoints where *K1* is the vertex keypoint). Default is in radians.
Nearest to Entity	
NNEAR(*N*)	Selected node nearest node *N*.
KNEAR(*K*)	Selected keypoint nearest keypoint *K*.
ENEARN(*N*)	Selected element nearest node *N*. The element position is calculated from the selected nodes.
Areas	
AREAND(*N1*, *N2*, *N3*)	Area of the triangle with vertices at nodes *N1*, *N2*, and *N3*.
AREAKP(*K1*, *K2*, *K3*)	Area of the triangle with vertices at keypoints *K1*, *K2*, and *K3*.
ARNODE(*N*)	Area at node *N* apportioned from selected elements attached to node *N*. For 2-D planar solids, returns edge area associated with the node. For axisymmetric solids, returns edge surface area associated with the node. For 3-D volumetric solids, returns face area associated with the node. For 3–D, select all the nodes of the surface of interest before using ARNODE.
Normals	
NORMNX(*N1*, *N2*, *N3*)	X-direction cosine of the normal to the plane containing nodes *N1*, *N2*, and *N3*.
NORMNY(*N1*, *N2*, *N3*)	Y-direction cosine of the normal to the plane containing nodes *N1*, *N2*, and *N3*.
NORMNZ(*N1*, *N2*, *N3*)	Z-direction cosine of the normal to the plane containing nodes *N1*, *N2*, and *N3*.

"Get Function" Summary

Entity Status Get Function	Description
NORMKX(*K1, K2, K3*)	X-direction cosine of the normal to the plane containing keypoints *K1, K2*, and *K3*.
NORMKY(*K1, K2, K3*)	Y-direction cosine of the normal to the plane containing keypoints *K1, K2*, and *K3*.
NORMKZ(*K1, K2, K3*)	Z-direction cosine of the normal to the plane containing keypoints *K1, K2*, and *K3*.
Connectivity	
ENEXTN(*N, LOC*)	Element connected to node *N*. *LOC* is the position in the resulting list when many elements share the node. A zero is returned at the end of the list.
NELEM(*E*,*NP0S*)	Node number in position *NP0S* (1--20) of element *E*.
NODEDOF(*N*)	Returns the bit pattern for the active DOFs at the specified node. bit 0 is UX, bit 1 is UY,... bit 5 is ROTZ bits 6,7,8 are AX,AY,AZ bits 9,10,11 are VX,VY,VZ bit 18 is PRES, bit 19 is TEMP, bit 20 is VOLT, bit 21 is MAG bit 24 is EMF, bit 25 is CURR For a node with UX,UY,UZ the return value will be 7 (bits 0,1,2) For a node with UX,UY,UZ,ROTX,ROTY,ROTZ the return value will be 63 (bits 0,1,2,3,4,5)
Faces	
ELADJ(*E, FACE*)	For 2-D planar solids and 3-D volumetric solids, element adjacent to a face (*FACE*) of element *E*. The face number is the same as the surface load key number. Only elements of the same dimensionality and shape are considered. A -1 is returned if more than one is adjacent.
NDFACE(*E, FACE, LOC*)	Node in position *LOC* of a face number *FACE* of element *E*. The face number is the same as the surface load key number. LOC is the nodal position on the face (for an IJLK face, LOC=1 is at node I, 2 is at node J, etc.)
NMFACE(*E*)	Face number of element *E* containing the selected nodes. The face number output is the surface load key. If multiple load keys occur on a face (such as for line and area elements) the lowest load key for that face is output.
ARFACE(*E*)	For 2-D planar solids and 3-D volumetric solids, returns the area of the face of element *E* containing the selected nodes. For axisymmetric elements, the area is the full (360 degree) area.
Degree of Freedom Results	
UX(*N*)	UX structural displacement at node *N*.
UY(*N*)	UY structural displacement at node *N*.
UZ(*N*)	UZ structural displacement at node *N*.
ROTX*(N*)	ROTX structural rotation at node *N*.
ROTY(*N*)	ROTY structural rotation at node *N*.
ROTZ(*N*)	ROTZ structural rotation at node *N*.
TEMP(*N*)	Temperature at node *N*. For **SHELL131** and **SHELL132** elements with KEYOPT(3) = 0 or 1, use TBOT(*N*), TE2(*N*), TE3(*N*), . . ., TTOP(*N*) instead of TEMP(*N*).
PRES(*N*)	Pressure at node *N*.

"Get Function" Summary

Entity Status Get Function	Description
VX(*N*)	VX fluid velocity at node *N*.
VY(*N*)	VY fluid velocity at node *N*.
VZ*(N*)	VZ fluid velocity at node *N*.
ENKE(*N*)	Turbulent kinetic energy (FLOTRAN) at node *N*.
ENDS(*N*)	Turbulent energy dissipation (FLOTRAN) at node *N*.
VOLT(*N*)	Electric potential at node *N*.
MAG(*N*)	Magnetic scalar potential at node *N*.
AX(*N*)	AX magnetic vector potential at node *N*.
AY(*N*)	AY magnetic vector potential at node *N*.
AZ(*N*)	AZ magnetic vector potential at node *N*.
Returns information about the data base manager	
VIRTINQR(1)	Number of pages in core.
VIRTINQR(4)	Page size in integer words.
VIRTINQR(7)	Maximum number of pages allowed on disk.
VIRTINQR(8)	Number of read/write operations on page.
VIRTINQR(9)	Maximum record number on page.
VIRTINQR(11)	Maximum pages touched.
Returns the current value of ANSYS filtering keywords.	
KWGET(*KEYWORD*)	Returns the current value the keyword specified by *KEYWORD*. See the *ANSYS UIDL Programmer's Guide* for a list of keywords and values.

Character String Functions Strings must be dimensioned (see ***DIM**) as a character parameter or enclosed in single apostrophes ('char').

Functions which return a double precision value of a numeric character string.

VALCHR(*a8*)	*a8* is a decimal value expressed in a string.
VALOCT (*a8*)	*a8* is an octal value expressed in a string.
VALHEX(*a8*)	*a8* is a hex value expressed in a string.

Functions which return an 8 character string of a numeric value.

CHRVAL (*dp*)	*dp* is a double precision variable.
CHROCT (*dp*)	*dp* is an integer value.
CHRHEX(*dp*)	*dp* is an integer value.

Functions which manipulate strings: ***StrOut*** *is the output string (or character parameter)* ***Str1*** *and* ***Str2*** *are input strings. Strings are a maximum of 128 characters. (see* ****DIM****)*

StrOut = STRSUB(Str1, nLoc,nChar)	Get the nChar substring starting at character nLoc in Str1.
StrOut = STRCAT(Str1,Str2)	Add Str2 at the end of Str1.
StrOut = STRFILL(Str1,Str2,nLoc)	Add Str2 to Str1 starting at character nLoc.
StrOut = STRCOMP(Str1)	Remove all blanks from Str1
StrOut = STRLEFT(Str1)	Left-justify Str1
nLoc = STRPOS(Str1,Str2)	Get starting location of Str2 in Str1.
nLoc = STRLENG(Str1)	Location of last nonblank character
StrOut = UPCASE(Str1)	Upper case of Str1
StrOut = LWCASE(Str1)	Lower case of Str1

The following functions manipulate file names.

"Get Function" Summary

Entity Status Get Function	Description
Path String = JOIN ('directory','filename','extension')	Produces a contiguous pathstring. e.g. directory/filename.ext
Path String = JOIN ('directory','filename')	Produces a contiguous pathstring. e.g. directory/filename
SPLIT('PathString', 'DIR')	Produces a separate output of the directory from the pathstring.
SPLIT('PathString', 'FILE')	Produces a separate output of the complete filename (with extension) from the pathstring.
SPLIT('PathString', 'NAME')	Produces a separate output of the filename from the pathstring.
SPLIT('PathString', 'EXT')	Produces a separate output of the file extension from the pathstring.

Index

Symbols

A

C

D

E

F

G

H

I

L

T

U

V

W