UIDL Programmer's Guide

ANSYS Release 10.0

002186
August 2005

UIDL Programmer's Guide

ANSYS Release 10.0

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
001384	ANSYS 5.7.1	May 2001
001513	ANSYS 6.1	March 2002
001974	ANSYS 8.1	April 2004
002098	ANSYS 9.0	November 2004
002186	ANSYS 10.0	August 2005

Copyright and Trademark Information

Disclaimer Notice

U.S. GOVERNMENT RIGHTS

Third-Party Software

Published in the U.S.A.

Table of Contents

List of Figures

List of Tables

Preface

Conventions Used in This Guide

This guide uses the following typographic conventions to indicate various types of information:

Convention	Indicates
COMMAND	ANSYS commands. These are shown as uppercase, bold text (for example, **K**, **DDELE**, and so on). In the online documentation, these provide hyperlinks to the appropriate command reference information.
Menu > Item	Menu paths (sometimes referred to as GUI paths). These are shown as bold text with mixed-case, separated by angle brackets ">". An angle bracket indicates a branch to a new menu item.
path/filename.ext	File names, which may or may not include directory paths. These are shown as lower-case, bold text, unless case is significant. Examples are shown with the UNIX directory separator character "/" (slash); if you are using a Microsoft Windows system, use "\" (backslash) as your directory separator character.
ARGUMENT	Arguments for numeric values (such as *VALUE, INC, TIME*) in command syntax. These are shown as upper-case italic text. On some commands, non-numeric convenience labels (for example, ALL and P) can also be entered for these arguments.
Argument	Arguments for alphanumeric values (for example, *Lab* or *Fname*) in command syntax. These are shown in mixed-case, italic letters. The guide also uses italic text for emphasis.
ANSYS Guide Title	The name of an ANSYS manual.
`command,arg1,arg2`	Command input listings, ANSYS output listings, and text that a user enters are shown in fixed-width font.
Note--	Information that supplements the main topic being discussed, such as important tips or guidelines.
Caution:	Actions or situations that could cause problems, unexpected ANSYS behavior, or unexpected results.
Warning	Actions or situations that can shut down ANSYS, damage files, cause loss of data, and so on.

About the Programmer's Guide Set

The ANSYS programmer's guide set provides information about the various programming interfaces available to customers. These manuals assume that you have at least a basic knowledge of programming (a working knowledge of Fortran 90 would be very helpful). The set of four manuals includes:

The APDL Programmer's Guide

This guide was designed for ANSYS users that have some programming skills and wish to tap the power of the ANSYS Parametric Design Language (APDL) to increase the productivity. APDL is a scripting language that is very similar to Fortran 90. The guide describes how to define parameters (variables), how to create macro programs using APDL, how to use APDL for simple user interaction, how to encrypt an APDL macro, and how to debug an APDL macro.

The UIDL Programmer's Guide

The UIDL Programmer's Guide covers the User Interface Design Language (UIDL) including how to modify or construct menus, dialogs and online help from within ANSYS.

Guide To ANSYS User Programmable Features

ANSYS provides a set of Fortran 90 functions and routines that are available to extend or modify the program's capabilities. Using these routines requires relinking the ANSYS program, resulting in a custom version of ANSYS. ANSYS provides an external commands capability which you can use to create shared libraries available to ANSYS (either from ANSI standard C or Fortran 90). you can use this feature to add custom extensions to ANSYS without the need to rebuild the ANSYS executable.

Guide to Interfacing with ANSYS

This guide describes a group of utilities as well as a set Fortran 90 routines that you can use to directly access the ANSYS database. You can also use these capabilities to access data in any of the binary files that ANSYS writes or uses.

Chapter 1: What is UIDL?

1.1. What Is the ANSYS UIDL?

The ANSYS User Interface Design Language (UIDL) is a programming language that lets you customize many components of the ANSYS Graphical User Interface (GUI). The configurable components include:

- Items on the ANSYS Main Menu
- Dialogs, including picking dialogs
- Online help

Customizing the GUI lets you modify many ANSYS menus and dialogs and integrate locally-developed programs into the ANSYS environment.

1.2. The Structure of UIDL

Each UIDL program consists of a control file header and a series of "building blocks." A building block is a series of UIDL commands, used to create a component of the GUI. There are two types of building blocks:

- Menu blocks, for creating menus
- Function blocks, for creating dialogs

The UIDL processor reads the control file, inserts ANSYS indexing information (which relates the building blocks to the ANSYS GUI) into the control file, and builds the GUI.

To customize the ANSYS GUI with UIDL, perform the following steps:

1. Create at least one new control file and save it to a subdirectory.
2. Copy the control file to the directory where your copy of the **menulist100.ans** file resides.
3. Add the name of the new local control file to the **menulist100.ans** file.
4. Run ANSYS to see if the changes you made appear in the GUI.
5. Make any changes to the original in the subdirectory and copy it to the same directory as you did in Step 2.

1.2.1. Control Files

A control file consists of a control file header and at least one building block. A control file name always ends with the extension **.GRN**. ANSYS uses control files to build all the standard dialogs and menus. The control files are stored in the **/ansys100/docu** directory. These files are extremely useful in helping you get started with UIDL since they contain many examples of how to code menus, and dialogs in UIDL. Here is a list of some of the standard ANSYS control files and the components of the GUI they build:

UIFUNC1.GRN	GUI Functions
UIFUNC2.GRN	GUI Functions
UIMENU.GRN	ANSYS Menus

Note — *Never* modify these files! Modifying the standard ANSYS control files will make the ANSYS GUI fail. Before you do any UIDL programming, copy these files to a subdirectory of your working directory, then copy the blocks out of the files you want to modify.

1.2.1.1. Control File Header

A control file header is a set of specific UIDL commands that defines information about the control file for the UIDL processor. A line in a control file cannot exceed 80 characters in length. All header commands begin with a colon (:). A control file header contains the following four commands in this order:

```
:F NEWMENU.GRN
:D Modified on %E%, defines new menus for ANSYS GUI
:I      0,       0,       0
:!
```

Here are brief descriptions of these commands:

:F *Filename*	Required; name of the control file, must appear on line 1.
:D*String*	Required; control file description, must appear on line 2. If you maintain source under a system such as SCCS, strings enclosed with percent signs (such as %*E*%) are expanded according to conventions defined by the source control system. If you do not use SCCS, insert a date instead of %*E*%.
:I 0, 0, 0	Required; space holder for ANSYS indexing, must appear on line 3. The 0s (zeros) must be in columns 9, 18, and 27 and must be separated by commas.
:!	Optional but encouraged; separates the control file header from the blocks, must appear on line 4.

For more information on the commands, see Appendix A:, UIDL Command Dictionary.

When UIDL processes an edited control file, the 0s and commas are replaced with indexing information. If you modify an indexed control file, you must reinsert the **:I** command with the 0s and commas in their proper places.

Note — Keep a clean copy of every control file in a subdirectory. Consider creating a one-line file that consists of **:I** and the 0s and commas in their correct locations. Whenever you create a new control file, you can insert the one-line file and ensure that the location of the 0s (zeros) is always correct.

Once you start ANSYS after having created new control files, ANSYS adds additional information to the end of the control file. ANSYS inserts **:X INDEX ADDED BY ANSYS** at the end of each control file. The name of each block appears, followed by an index number. If you ever modify an indexed control file, you must delete everything after the final **:E END** command of the control file.

1.2.2. Building Blocks

Building blocks follow the control file header in the control file. Building blocks can be menu blocks or function blocks.

As you build new blocks, create at least one control file for menus and one for functions. Store a clean copy of each control file in a subdirectory. Always test each block separately before adding it to one of your final control files--set up a **TEST.GRN** file as described in Appendix D:, Testing and Troubleshooting. Store the final versions of your control files in your working directory.

1.2.2.1. Menu Blocks

The menu blocks control the organization and content of the ANSYS Main Menu in the GUI. The standard ANSYS menu blocks are all contained in the **UIMENU.GRN** file. Menu blocks are discussed in detail in Chapter 2, "Modifying Menu Blocks".

1.2.2.2. Function Blocks

Function blocks control the following:

- Layout and content of the configurable dialogs and prompts
- Picking operations
- Hidden operations which build and pass commands from the GUI to ANSYS

The standard ANSYS dialogs are stored in the **UIFUNC1.GRN** and the **UIFUNC2.GRN** files. Function blocks are discussed in detail in Chapter 3, "Modifying Function Blocks".

1.3. Overview of Building Blocks

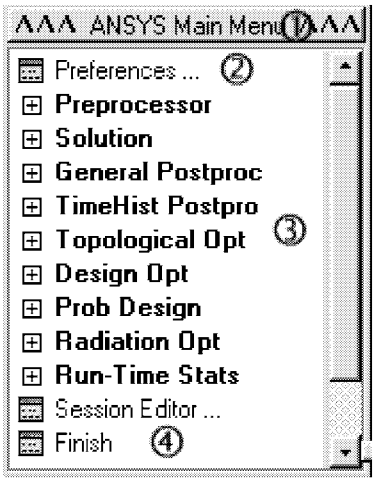
Each building block consists of three sections:

- Header Section
- Data Controls Section
- Ending Section

Each type of building block has different data controls. Each control file typically contains multiple building blocks.

Figure 1.1: "A Control File Containing the ANSYS Main Menu Building Block" shows a sample control file containing a menu block for an ANSYS GUI menu. This menu block builds the ANSYS Main Menu. The numbers relate portions of the code to some of the options on the menu.

Figure 1.1 A Control File Containing the ANSYS Main Menu Building Block



Control File Header

```
:F NEWMENU.GRN
:D Menu Control File
:I      0,       0,       0
:!
```

Block Header Section

```
:N MenuRoot
:S      0,      0,      0
:T Menu
:A Main Menu                  ❶
```

Data Controls Section

```
:D ANSYS ROOT MENU
Fnc_Preferences               ❷
Sep_
Men_Preproc
Men_Solution
Men_GenlPost
Men_TimePost
Sep_                          ❸
K_LN(alpha)
Men_DesOpt
K_LN(ALPHA)
Men_DesOpt_al
Men_Aux12
Men_RunStat
Sep_
Fnc_FINISH                    ❹
K_LN(UTILMENU)
Men_UtilMenu
```

Ending

```
:E END
```

Note — Menu names are prefixed by the **Men_** string. The only exception to this is the MenuRoot menu depicted above.

1.3.1. The Header Section

All header commands begin with a colon (**:**). Each header section must begin with a **:N** command which defines the unique, internal name of the building block. The next command, **:S,** is a space holder for the index line. This command must have zeros, separated by commas, in columns 9, 16, and 23. Consider creating a one-line file that consists of **:S** and the 0s and commas in their correct locations.

The **:T** indicates the type of building block: **Menu**, or **Cmd**. Optional UIDL commands, such as **:A** (menu heading) and **:D** (block description), can also be in the header. The header section is processed as soon as the building block is entered by the UIDL processor.

1.3.2. The Data Controls Section

This section must include at least one data control. The data controls used depend on the type of building block defined by the **:T** line. In menu blocks, the data control section contains calls to existing menu blocks (**Men_***String*) and to function blocks **Fnc_***String*). In function blocks, the data control section defines the contents of a dialog. The discussions of each building block type in the following chapters describe which specific data controls to use in each type of block.

1.3.3. Ending Line

A block must always end with the **:E END** command. At least one separator (**:!** command) should appear between blocks.

1.4. General Guidelines for Control Files

When creating a control file, always be aware that the UIDL processor inserts additional information in each control file. The processor overwrites all **:I** and **:S** command lines with indexing information, and adds the **:X** command and the internal name of each block to the end of the control file. It is critical that you always save an extra copy of any new control file and block, and store them in a directory other than your working directory or your login directory. Whenever you modify a control file, you *must* start with a clean copy, not one that ANSYS has modified.

An error in a block can cause lines to be deleted in the control file or can cause erratic behavior when the UIDL processor indexes it.

For easier readability, separate each block with the **:!** command and use uppercase characters for all ANSYS commands.

1.5. The menulist100.ans File

In order for ANSYS to incorporate your changes into the GUI, you need to add the name of the any new control file to the **menulist100.ans** file. This file contains the pointers to all ANSYS control files, which in turn contain the GUI building blocks. ANSYS looks for **menulist100.ans** using the following search path order:

1. The current working directory
2. The user's home login directory
3. The **docu** directory

If **menulist100.ans** is not found, an error condition occurs. ANSYS reads the contents of **menulist100.ans** sequentially, checking to see if each control file has been indexed. If a control file has not yet been indexed, the UIDL processor does the indexing and updates the control file automatically. Indexing tells ANSYS where each of the building blocks fits into the structure of the GUI.

Each time you create a new control file, you must update the **menulist100.ans** file. The **menulist100.ans** file containing pointers to your customized control files should be in your working directory or home directory. The **menulist100.ans** file allows a maximum of 20 pointers. If ANSYS finds multiple blocks with the same name during the indexing process, the GUI uses the last block found. Always insert pointers to your control files at the end of the **menulist100.ans** file.

Here is a sample **menulist100.ans** file, with one new control file added:

```
/ansys100/docu/UIMENU.GRN
/ansys100/docu/UIFUNC1.GRN
/ansys100/docu/UIFUNC2.GRN
NEWMAIN.GRN
```

The ANSYS GUI information defined by the **NEWMAIN.GRN** file takes precedence over the GUI information defined in the standard ANSYS control files because it appears last in the **menulist100.ans** file.

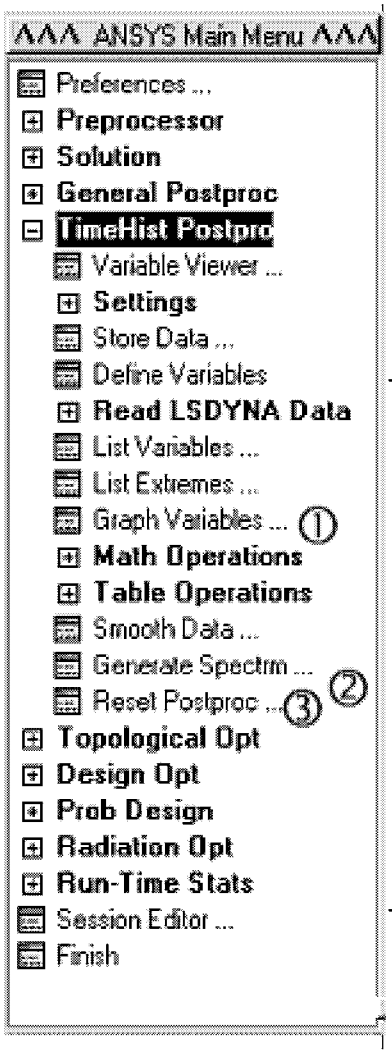
Chapter 2: Modifying Menu Blocks

2.1. Creating Menu Blocks

You can use UIDL to modify existing ANSYS menus or to create new ones. This chapter documents how to modify or create menus.

Figure 2.1: "Menu Block for the TimeHist Postproc Menu" shows the menu block for the **TimeHist Postproc** menu. The numbers relate portions of the code to some of the options on the menu.

Figure 2.1 Menu Block for the TimeHist Postproc Menu



Header

```
:N Men_TimePost
:S   0,        0,     0
:T Menu
:A TimeHist Postpro
:C )/nopr
:C )*get,_z1,active,,routin
:C )*if,_z1,ne,36,then
:C )_z2='/POST26'
:C )*else
:C )_z2=')!'
:C )*endif
:C %_z2%
:C )/go
:D Time-History Postprocessor (POST26)
```

Data Controls

```
Men_p26Settings
Fnc_STORE
Fnc_p26DefineVar
Sep_
Men_DYNA_Read
Sep_
Fnc_PRVAR_p26
Fnc_EXTREM
Fnc_PLVAR_p26      ❶
Sep_               ❷
Men_p26Oper
Men_p26Table
Fnc_RESP
Men_p26Emag
K_LN(lsdyna)
Sep_
Fnc_RPSD
Fnc_COPSD
Sep_
Fnc_RESET_p26      ❸
```

Ending

```
:E END
```

This example also shows the use of a simple APDL routine in the header section, which is used to determine the current ANSYS routine, and, if it is not in /POST26, it will enter the Time-History Postprocessor. See the *ANSYS APDL Programmer's Guide* for more information. The **:C** UIDL command calls ANSYS commands, which include APDL routines, user-programmable functions and external libraries.

The data controls section lists all commands which will appear on a menu. Note that on a menu, each reference to a submenu ends in a greater-than sign, and each reference to a dialog ends in ellipses.

2.1.1. Menu Block Header Section Commands

The header section of the menu block uses the UIDL commands which begin with a colon (:). The following commands are valid in the header section of a menu block:

:!	Separates menu blocks and designates comments.
:N Men_*String*	Required; defines the internal name of the menu block.
:S 0, 0, 0	Required; space holder for ANSYS indexing. The 0s (zeros) must be in columns 9, 16, and 23 and must be separated by commas.
:T Menu	Required; defines the type of the block. To define a menu block the string Menu must appear after the **:T** command.
:A*String*	Required; defines the title of the menu block as it appears on a submenu. When the current menu block is called by another menu block, it displays the title of the menu and appends a > to the string. *Note*--Items listed on a menu under a subhead should be indented by three spaces.
:D*String*	Optional; describes the menu block.
:C*Command String*	Optional; executes ANSYS commands, APDL, user-defined functions or external library calls when this menu block is entered.
:K*Keyword Logic*	Optional; filters this menu block based on keywords. See Appendix A:, UIDL Command Dictionary for a list of ANSYS keywords.
:P*Product Code Logic*	Optional; filters this menu block based on product codes. See Appendix Appendix B:, ANSYS Keywords for a list of product codes.

2.1.2. Menu Block Data Controls Section

The data control section of the menu block requires at least one line of information. While you can include many data control commands, you should avoid making a menu too long. When the menu length exceeds the boundaries of the screen, the user cannot access all parts of the menu. Break long menus down into smaller menu blocks. The following data control UIDL commands are valid:

Men_*String*	Adds the menu block named by *String* to the menu. The *String* used in this command must match a *String* defined by an **:N** command in another menu block.
Fnc_*String*	Adds the function block named by *String* to the menu. The *String* used in this command must match a *String* defined by an **:N** command in a function block.
Sep_	Inserts a separator line on the menu between commands.
String	Inserts the text to be used as a menu title. *Note*--In the menu block header, the contents of the **:A** string should be indented by three spaces.

2.1.3. Menu Block Ending Line

The **:E END** command always appears in the ending section of every block.

2.2. General Guidelines for Writing Menu Block Control Files

Menu blocks should be stored in their own control files. If you need to modify a control file that ANSYS has already processed, you must reset all the index strings (the lines beginning with **:I** and **:S**) to the sequence of space-holding 0s. You must also delete all the information following the last **:E END** command near the end of the control file. A line in a control file cannot exceed 80 characters in length.

> *Note* — Always copy material from the **UIMENU.GRN** file to create your own menu control files. *Never* modify this file.

If you want to create a completely new menu structure, you must include the menu block named `MenuRoot` and retain its internal name (`MenuRoot`).

Hide a function in a menu if the user explicitly turned it off through the Preferences dialog. If no preference was set, the user should see the item dimmed if its functionality is not available.

Chapter 3: Modifying Function Blocks

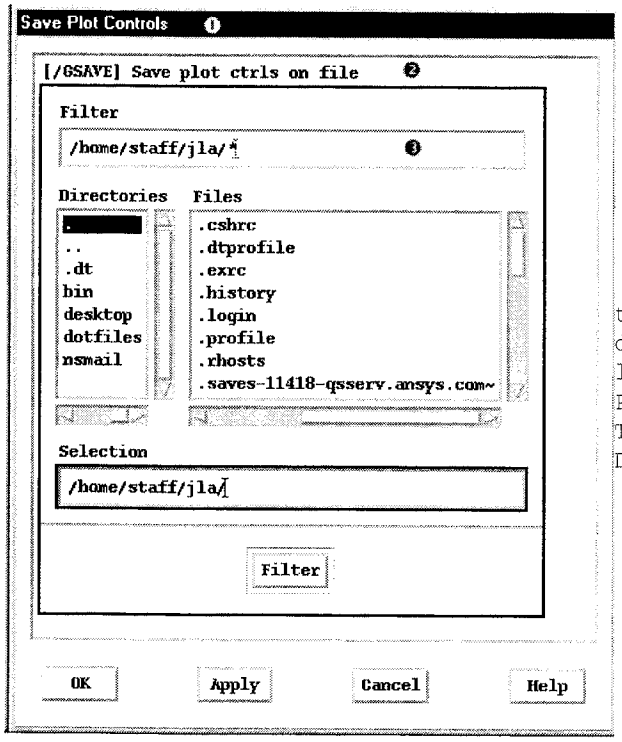
3.1. Types of Function Blocks

Menu blocks are little more than lists of calls to menus and dialogs, so they are very straightforward to build and modify. Function blocks, however, include extensive functionality for accepting various types of input from users. There are several types of function blocks:

Dialog

A dialog accepts input from the user and processes it according to the ANSYS commands included in the function block. Data controls build most dialogs with an absolute minimum of UIDL coding. Figure 3.1: "Function Block for the Save Plot Controls Dialog" shows a standard ANSYS dialog with its associated function block, which uses the **/GSAVE** command to save plot controls to a selected directory.

Figure 3.1 Function Block for the Save Plot Controls Dialog



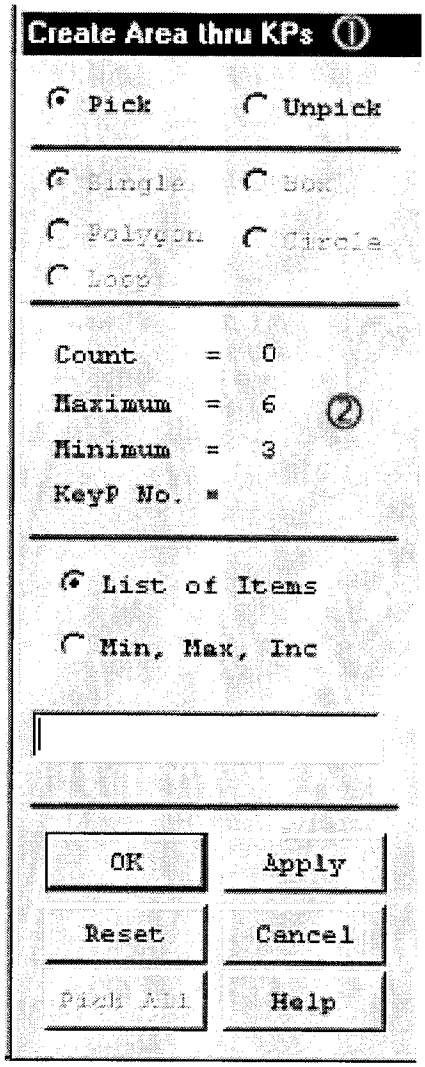

```
Fnc_Pl_/GSAVE
     0,     0,     0
 Cmd
 )! Fnc_Pl_/GSAVE
 )/NOPR
 )*SET,_zx,'*'
 )/GO
 Hlp_C_/GSAVE
 Save Plot Ctrls
 Save Plot Controls  ❶
t_H
d_/GSAVE
ld_2
Prm_[/GSAVE] Save plot ctrls on file ❷
Typ_FILE      ❸
Def_*PAR(_zx)
 END
```

When building the data control section for a dialog, use at least one **Typ_** data control and do not use any **Inp_P** controls.

Picking box

A picking box lets the user select keypoints, lines, volumes, nodes (etc.) from a model. Here is a standard ANSYS picking box, which uses the ANSYS **A** command to create areas through selected keypoints, and requires that at least three keypoints be selected:

Figure 3.2 Function Block for a Create Area thru KPs Picking Box




```
:N Fnc_A
:S      0,      0,      0
:T Cmd_P
:C )! Fnc_A
:A Through KPs
:D Create Area thru KPs ❶
:K #(PREP7)
:H Hlp_C_A
Inp_P
Cmd_A
 Fld_2
  Typ_KEYP
  Prm_[A]  Pick or enter
  keypoints defining the area
  Rub_53
  Min_3  ❷
  Cnt_s
:E END
```

When building the data control section of a picking box, include the **Inp_P** data control and at least one **Typ_** data control.

Hidden

A hidden function is performed without displaying a dialog or picking box. Figure 3.3: "Function Block for a Hidden Function" shows a sample hidden function block:

Figure 3.3 Function Block for a Hidden Function

```
:N Fnc_CirBD_3
:S      0,      0,      0
:T Cmd
:A Hidden (Change Element Color)
:C )! Fnc_CirBD_3
:C )/nopr
:C )*get,_z(30),graph,,number
:C )/pnum,mat,1
:C )/num,1
:C )eplot
:C )/num,_z(30)
:C )/go
:P (ELECMAG)
Inp_P
```

```
Cmd_)!
:E END
```

When building the data control section for a hidden function, include the **Inp_P** data control and at least one **Cmd_** data control with no **Typ_** data controls which require input from the user. To make it more readable include **:A** Hidden in the header section of the function block, and include some information about what the function block does.

3.1.1. Function Block Header Section Commands

The header section of the function block uses the UIDL commands which begin with a colon (**:**). The header section of the function block can include the following UIDL commands:

:!	Separates function blocks and designates comments.
:N Fnc_*String*	Required; defines the internal name of the function block.
:S 0, 0, 0	Required; space holder for ANSYS indexing. The 0s (zeros) must be in columns 9, 16, and 23 and must be separated by commas.
:T Cmd	Required; defines the type of the block. To define a function block the string Cmd must appear after the **:T** command.
:A*String*	Optional but strongly suggested; defines the name of the function as it will appear on a menu. If you omit this command, the internal name defined by the **:N** command will appear on menus. *Note*--Items listed on a menu under a subhead should be indented by three spaces.
:D*String*	Required; provides the heading for the dialog, which appears in the title bar of the dialog window.
:C*Command*	Optional; executes ANSYS commands, APDL, user-defined functions or external library calls when this function block is entered.
:K*Keyword Logic*	Optional; filters this function block based on keywords. See Appendix B:, ANSYS Keywords for a complete list of keywords.
:P*Product Code Logic*	Optional; filters this function block based on product codes. See Appendix C:, ANSYS Product Codes for a complete list of product codes.
:H Hlp *_String*	Optional; defines the internal name of the help which documents the dialog. The procedure for naming and writing help is documented in Chapter 4, "Creating Help".

3.1.2. Function Block Data Control Section Commands

The data control section of the function block supports many data control commands. Each command has been grouped by functionality. For detailed descriptions of each command, see Appendix A:, UIDL Command Dictionary.

3.1.2.1. General Data Control Commands

!	Defines a comment line
Cal_	Calls another function block; must be used just before the **:E END** command
Cal_REFRESH	Forces a refresh of the lowest active side menu
Fmt_H	Forces a narrow dialog
Inp_ P	Defines picking boxes or hidden function blocks
Inp_NoApply	Suppresses the **Apply** button
K_LN	Sets keyword logic for the next line
P_LN	Sets product code logic for the next line
Pwr_	Switches **FLST** and **FITEM** on and off

Rmk_	Forces a rebuild of a dialog after an **Apply**
Typ_Def_	Defines a default value for a field which is not displayed
Typ_Lab	Displays label without an associated button or entry field

3.1.2.2. Command Controls

Cmd_	Initiates a command sequence
K_CM	Sets keyword logic for this command
P_CM	Sets product code logic for this command

3.1.2.3. Field Controls

General Controls

Dlm_	Delimits fields
Def_	Sets initial default value for the field
Fld_	Defines the field number, typically related to the **Cmd_** preceding it
K_FL	Sets keyword logic for this field
P_FL	Sets product code logic for this field
Prm_	Defines label or prompt to use with this field

Picking Controls

Cnt_	Sets maximum number of items, except an apply is not done
Max_	Sets maximum number of items
Min_	Sets minimum number of items
Pcn_	Defines ordered or unordered set
Pdp_	Permits duplicate entity picking
Pfm_	Controls how picked items are processed
Rub_	Defines the type of rubber banding to use
Sel_	Defines type of selection allowed
Typ_*Entity*	Gets the numbers of the specified entities
Typ_Resu	Gets the results for a specified entities
Typ_XYZ	Gets the coordinates of a point
Typ_XYZ_SCREEN	Gets the screen coordinates
Typ_XYZ_WP	Gets the working plane coordinates of a point

Listing Controls

Bnd_ *LOWER,UPPER*	Creates bounds for **Typ_MLis**
Typ_Color	Creates an option button of available colors
Typ_Idx	Creates side-by-side scrolling lists from the subsequent **Idx_** controls
Typ_Lis _OptionB	Creates an option button list
Typ_Lis _RadioB	Creates a radio button list
Typ_Lis	Creates a single selection scrolling list
Typ_MLis	Creates a multiple selection scrolling list

Lis_*item,value*	Creates items in the list
Lis_*READ,*item*	Defines a list of items from a database

File Controls

Typ_File	Creates a file selection box with filter, directory and file names
Typ_File_Inline	Creates a text entry field for passing a filename to ANSYS

Numerical Controls

Typ_Int	Creates a single input field for integer data
Typ_Int2	Creates two input fields for integer data
Typ_Int3	Creates three input fields for integer data
Typ_Real	Creates a single input field for real number data
Typ_Real2	Creates two input fields for real number data
Typ_Real3	Creates three input fields for real number data

Character Controls

Typ_Char	Creates an input field for character data

Logical Controls

Typ_Logi	Creates a toggle button for off and on labels

3.1.3. Function Block Ending Line

The **:E END** command always appears in the ending section of every block.

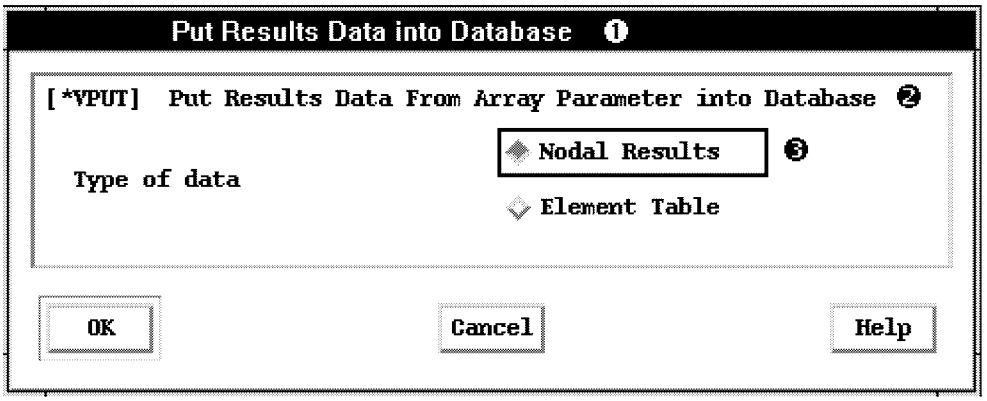
3.2. General Guidelines for Writing Function Block Control Files

Function blocks should be stored in their own control files. If you need to rebuild a new control file from an existing one, you must reset all the index strings (the lines beginning with **:I** and **:S**) to the sequence of space-holding 0s. You must also delete all the indexing information following the last **:E END** command near the end of the control file. *Always* keep a clean backup copy of every function block.

> *Note* — Always copy material from the **UIFUNC1.GRN** or **UIFUNC2.GRN** files to create your own function control files. *Never* modify these files.

Here is an example of a function block and the dialog it will produce:

Figure 3.4 The Function Block for the Put Results Data in Database Dialog



Header

```
:N Fnc_*VPUT
:S      0,     0,     0
:T Cmd
:C )!  Fnc_*VPUT
:A Put Array Data
:D Put Results Data into Database ❶
:H Hlp_C_*VPUT
```

Data Controls

```
Inp_NoApply
Cmd_)!
 Fld_0
  Typ_Lab
❷ Prm_[*VPUT]  Put Results Data From Array Parameter into Database
 Fld_2
  Prm_ Type of data
  Typ_LIS_RADIOB   ❸
  LIS_Nodal Results   ,0
  LIS_Element Table   ,1
Cal_Fnc_*VPUT_node,2,EQ,0
Cal_Fnc_*VPUT_etab,2,EQ,1
```

Ending

```
:E End
```

Some of the UIDL commands behave differently in a dialog block than in a menu. The **:D** command, for example, is used to insert text into the title bar of the dialog. There are some additional header commands for function blocks, such as the **:H** command. The **:H** command is used to associate an online help file with the dialog's Help button. The **Cal**_Fnc_ command is used rather than the **Cmd**_ command since ***VPUT** is called with different arguments depending on the type of data the user selects.

Here is a list of guidelines to help you write function blocks:

- Any ANSYS command name must be all uppercase and enclosed in brackets.
- Use the same case for field names as in the *ANSYS Commands Reference*.
- Use the **Typ_Sep** control to vertically separate logical groups within a dialog (for example commands, options, and so on).
- Use "white" space as a horizontal separator, not characters that is *, -, #, and so on).
- When a prompt is longer than one line, use a "-" dash at the end of the first line and at the beginning of the second line.
- Capitalize the title heading-style and be sure it describes the whole dialog.
- Avoid using a scroll bar in a dialog--limit the vertical height appropriately.
- Do not use picking inside a dialog. It does not exhibit clean behavior.
- *Do not* document the commands within the dialog. Keep field explanations brief. Use online help to document commands. See Chapter 4, "Creating Help" for more information.

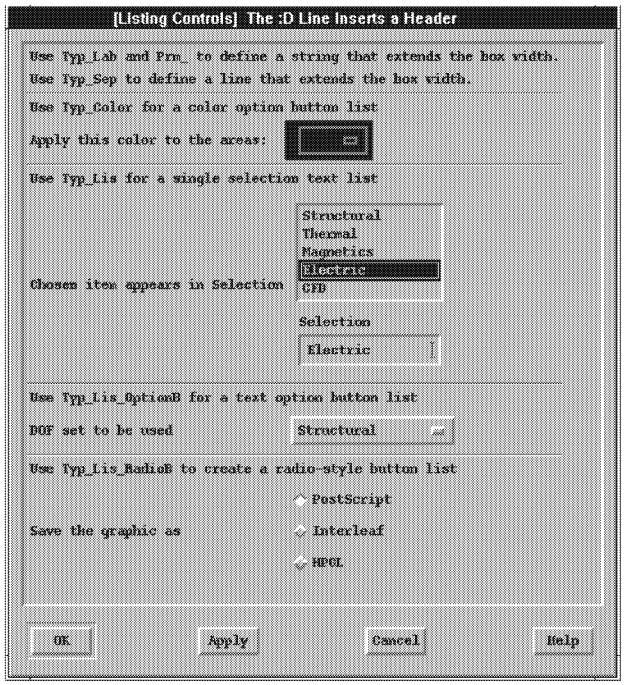
- Add a line containing the text **:C**)!Fnc_*String* to the header section of the function block. To write the name of the function block to the log file, type **KEYW**,QALOGKEY,1 into the ANSYS Input Window for debugging purposes.
- An error in a function block can cause lines to be deleted in the control file when the UIDL processor indexes it.
- For better readability, separate each function block with a **:!** line.
- You can use the **K_** and **P_** commands together, but you can use only one of each (with the exception of **K**_LN).
- If a dialog's contents will change after an Apply operation (for example, ***READ** for a list), place an **Rmk_** control at the top of the function's data control section.
- A line in a control file cannot contain more than 80 characters.

3.3. Sample Dialogs and Function Blocks

Figure 3.5: "Function Block for Creating a Dialog with Single-Selection Lists" through Figure 3.8: "Function Block for Creating a Dialog with Data Entry Fields" display UIDL code for function blocks and the dialogs or picking boxes they create.

Figure 3.5 Function Block for Creating a Dialog with Single-Selection Lists

```
:N Fnc_UIDL_001
:S      0,      0,      0
:T Cmd
:A List Ctrl Dialg
:D [Listing Controls]
  The :D Line
   Inserts a Header
:H Hlp_New_Fun
Cmd_/COM
 Fld_0
  Typ_Lab
  Prm_Use Typ_Lab and Prm_ to
 define a string that extends
 the box width.
 Fld_0
  Typ_Lab
  Prm_Use Typ_Sep to define a
 line that extends the box
 width.
 Fld_0
  Typ_Sep
 Fld_0
  Typ_Lab
  Prm_Use Typ_Color for a color option
button list
```

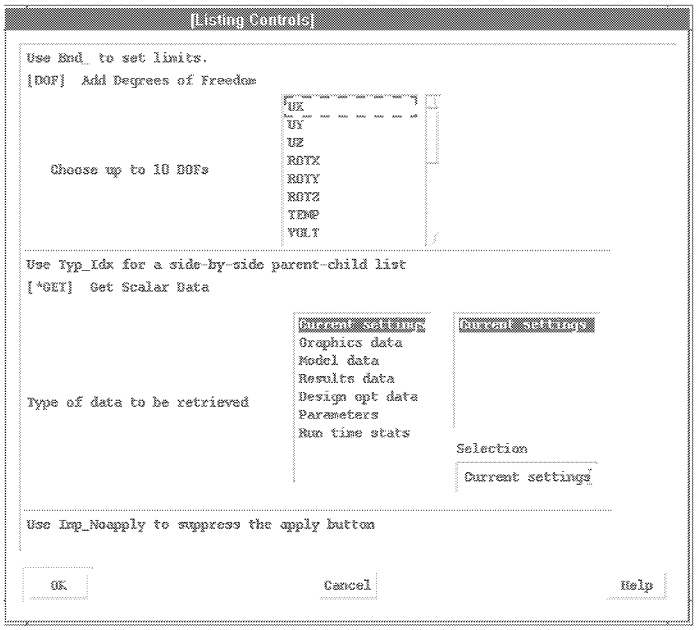



```
   Fld_2
     Prm_Apply this color to the areas:
     Typ_Color
    Fld_0
     Typ_Sep
    Fld_0
     Typ_Lab
     Prm_Use Typ_Lis for a single selection text list
    Fld_3
     Typ_Lis
      Lis_Structural,1
      Lis_Thermal,2
```

```
      Lis_Magnetics,3
      Lis_Electric,4
      Lis_CFD,5
     Prm_Chosen item appears in Selection
    Fld_0
     Typ_Sep
   Fld_0
     Typ_Lab
     Prm_Use Typ_Lis_OptionB for a text option button list
    Fld_4
     Prm_DOF set to be used
     Typ_Lis_OptionB
      Lis_Structural,UX,UY,UZ
      Lis_Electric,AX,AY,AZ
   Fld_0
     Typ_Sep
   Fld_0
     Typ_Lab
     Prm_Use Typ_Lis_RadioB to create a radio-style button list
    Fld_5
     Prm_Save the graphic as
     Typ_Lis_RadioB
      Lis_PostScript,ps
      Lis_Interleaf,il
      Lis_HPGL,gl
   :E End
```

Figure 3.6 Function Block for Creating a Dialog with Multiple-Selection Lists

```
:N Fnc_UIDL_002
:S      0,     0,     0
:T Cmd
:A MList Ctrl Dialg
:D [Listing Controls]
Inp_Noapply
Cmd_DOF
 Fld_0
  Typ_Lab
  Prm_Use Typ_MLis for
multiple selection text.
  Prm_Use Bnd_ to set limits.
 Fld_0
  Typ_Lab
  Prm_[DOF]  Add Degrees of
Freedom
 Fld_2
  Prm_   Choose up to 10 DOFs
  Typ_MLis
  Bnd_2.0,10.0
```



Items for the Single-Selection List

```
      Lis_UX              ,UX
      Lis_UY              ,UY
      Lis_UZ              ,UZ
      Lis_ROTX            ,ROTX
      Lis_ROTY            ,ROTY
      Lis_ROTZ            ,ROTZ
      Lis_TEMP            ,TEMP
      Lis_VOLT            ,VOLT
      Lis_MAG             ,MAG
      Lis_AX              ,AX
      Lis_AY              ,AY
      Lis_AZ              ,AZ
      Lis_VX              ,VX
```

```
    Lis_VY              ,VY
    Lis_VZ              ,VZ
    Lis_PRES            ,PRES
    Lis_ENKE            ,ENKE
    Lis_ENDS            ,ENDS
Cmd_/COM
Fld_0
  Typ_Sep
Fld_0
  Typ_Lab
  Prm_Use Typ_Idx for a side-by-side parent-child list
Fld_0
  Typ_Lab
  Prm_[*GET]  Get Scalar Data Fld_2
  Prm_Type of data to be retrieved
  Typ_Idx
```

Items for the Indexed, Multiple-Selection List

```
  Idx_Current settings,Current settings,1
  Idx_Graphics data   ,Graphics data   ,2
  Idx_Model data      ,Nodes           ,3
  Idx_Model data      ,Elements        ,4
  Idx_Model data      ,Keypoints       ,5
  Idx_Model data      ,Lines           ,6
  Idx_Model data      ,Areas           ,7
  Idx_Model data      ,Volumes         ,8
  Idx_Model data      ,For selected set,9
  Idx_Model data      ,Coord systems   ,10
  Idx_Model data      ,Element types   ,11
  Idx_Model data      ,Real constants  ,12
  Idx_Model data      ,Material props  ,13
  Idx_Model data      ,Data tables     ,14
  Idx_Results data    ,Global measures ,15
 Idx_Results data    ,Nodal results   ,16
  Idx_Results data    ,Element results ,17
  Idx_Results data    ,Modal results   ,18
  Idx_Results data    ,Elem table data ,19
  Idx_Results data    ,Elem table sums ,20
  Idx_Results data    ,Elem table misc ,21
  Idx_Results data    ,Path operations ,22
  Idx_Results data    ,Other operations,23
  Idx_Results data    ,Time-hist var's ,24
  Idx_Design opt data ,Design opt data ,25
  Idx_Parameters      ,Parameters      ,26
  Idx_Run time stats  ,Run time stats  ,27
 Fld_0
  Typ_Sep
 Fld_0
Typ_Lab
  Prm_Use Inp_Noapply to suppress the apply button
:E End
```

Figure 3.7 Function Block for Creating a Dialog with a File Selection Box

```
:N Fnc_UIDL_003
:S      0,     0,     0
:T Cmd
:A File Ctrl Dialg
:D [File Controls]
Inp_Noapply
Cmd_/COM
 Fld_0
  Typ_Lab
  Prm_Use Typ_File to use a
 file selection box with
 filter, dir & filename
 Fld_2
  Typ_File
  Prm_File to write to
 Fld_0
  Typ_Sep
 Fld_0
  Typ_Lab
  Prm_Use Typ_File_Inline to
 pass a filename to ANSYS as
 a string
 Fld_3
  Typ_File_Inline
  Prm_Write file to
:E End
```

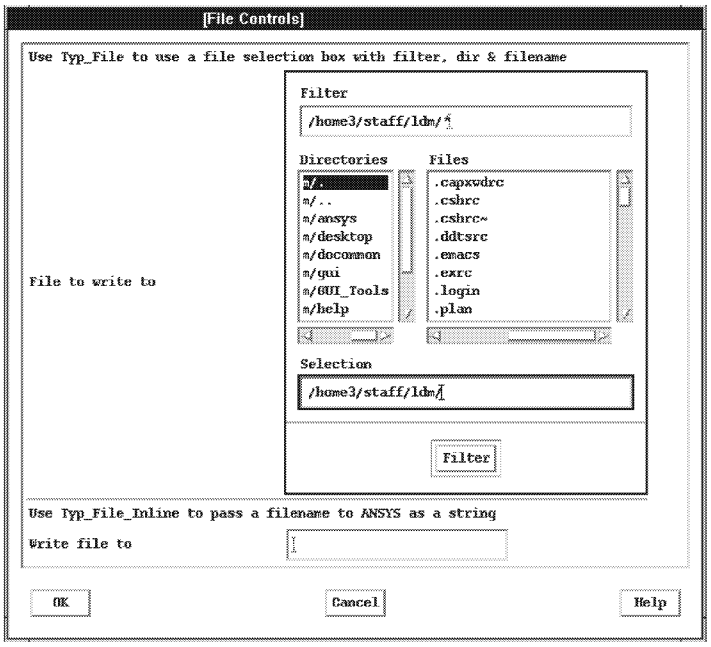
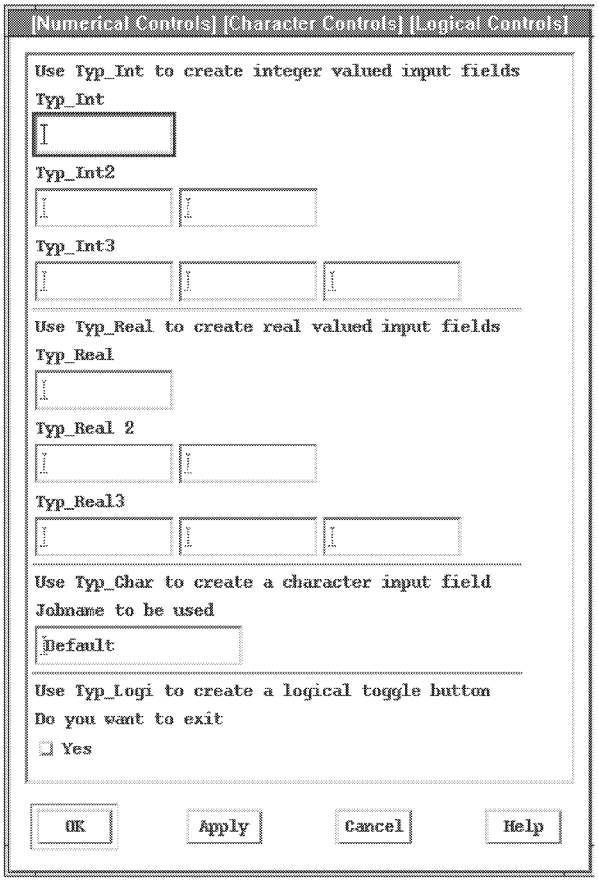


Figure 3.8 Function Block for Creating a Dialog with Data Entry Fields

```
:N Fnc_UIDL_004
:S      0,     0,     0
:T Cmd
:A Num Char Log Dialg
:D [Numerical Controls]
[Character Controls]
[Logical Controls]
Fmt_H
Cmd_/COM
 Fld_0
  Typ_Lab
  Prm_Use Typ_Int to create
 integer valued
input fields
 Fld_2
  Typ_Int
  Prm_Typ_Int
 Fld_3
  Typ_Int2
  Prm_Typ_Int2
 Fld_4
  Typ_Int3
  Prm_Typ_Int3
 Fld_0
  Typ_Sep
```




```
    Fld_0
     Typ_Lab
```

```
  Prm_Use Typ_Real to create real valued input fields
 Fld_4
  Typ_Real
  Prm_Typ_Real
 Fld_5
  Typ_Real2
  Prm_Typ_Real 2
 Fld_6
  Typ_Real3
  Prm_Typ_Real3
 Fld_0
  Typ_Sep
 Fld_0
  Typ_Lab
  Prm_Use Typ_Char to create a character input field
 Fld_7
  Typ_Char
  Def_Default
  Prm_Jobname to be used
 Fld_0
  Typ_Sep
 Fld_0
Typ_Lab
  Prm_Use Typ_Logi to create a logical toggle button
 Fld_8
  Typ_Logi,Yes,No
  Prm_Do you want to exit
:E End
```

3.4. Custom Dialogs

Custom dialogs are those dialogs which have been "hard" coded for GUI use. That is, they do not use the UIDL to create the layout and are usually invoked by issuing a variation of the ANSYS **STAT** command. The custom dialogs usually call other function blocks and/or ANSYS commands as a result of the user pressing a button in the custom dialog.

3.4.1. Maintaining the Interface Between UIDL and Custom Dialogs

To establish the relationships between the menu blocks and function blocks that call custom dialogs and the custom dialogs that call function blocks and/or commands, *always* include the following lines in the data controls of the building block that calls a custom dialog:

! Cust_Cal_Fnc_*Function_Name*
Lists the locations of the function in the ANSYS GUI.

! Cust_Cmd_*Command_Name*
Lists the locations of the command in the ANSYS GUI.

3.4.2. Example of a Custom Dialog

The following function block calls a custom dialog and then indicates the function block and ANSYS command that the custom dialog will call.

```
:N Fnc_*STAT_array
:S      0,      0,      0
:C )! Fnc_*STAT_array
:T Cmd
:A Hidden
:H Hlp_UI_Arry_Stat
Inp_P
```

Custom dialog entry

```
Cmd_)*STAT,,, ,,, ,,1
```

Function block called

```
! Cust_Cal_Fnc_ARRAY_ADD
```

ANSYS command to be executed

```
! Cust_Cmd_*VEDIT
:E End
```

It is possible for there to be more than one **! Cust_** line for a custom dialog and since they are comments they may be placed anywhere in the data control section of the block.

Chapter 4: Creating Help

4.1. Customizing ANSYS Help

Information on customizing your help system can be found in the appropriate section of the Operations Guide.

See the discussion on customizing help in the *ANSYS Operations Guide*.

Chapter 5: Advanced Topics

5.1. Using ANSYS Parameters for Data Flow

Advanced Topics

The nine scalar parameters, single array parameter, and single character array parameter listed below are "reserved" for use in the building blocks. They also must be the *only* parameters used in the building blocks. If you feel that using these parameters is confusing because they're not descriptive, create a correspondence table showing what the parameters map to.

```
_z1             scalar
_z2             scalar
_z3             scalar
_z4             scalar
_z5             scalar
_z6             scalar
_z7             scalar
_z8             scalar
_z9             scalar
_z(i,j,k)       array
_zc(i,j,k)      character array
```

You can use these parameters over and over again. Therefore, you cannot count on parameter values being preserved outside the function blocks in which they are defined and used, because other function blocks will be using the same parameter names.

When you use a scalar parameter, initialize it to prevent incorrect values.

When using array parameters, you must delete and then dimension the array parameter at the beginning of the function block. For example:

```
:C *DEL,_z(1)
:C *DIM,_z,,5,2
```

The ***DEL** command silently deletes parameters that begin with underscore (_), whether or not they exist or have been previously dimensioned.

You can also use the GUI command ***CSET** to pass a 32-character string. See *ANSYS Commands Reference* for more information.

To see the current status of hidden parameters, use the command ***STAT**,_prm. It will list all of the "_" parameters in use and their values.

5.2. Suppressing ANSYS Commands

To suppress the writing of an ANSYS command to the log file and to the internal database command log, insert a) (a right parenthesis) immediately before it. You can use this option in the GUI menu blocks to prevent the log file from becoming cluttered with commands (such as the ***GET** command) used to set up dialog box defaults, temporary UIDL underscore parameters, etc.

Use the) option in the building blocks to save users from seeing unfamiliar commands on their log files. *However, using this option can lead to CLASS3-type errors*. To ensure consistent usage and to avoid undesirable situations when creating building blocks, you *must* observe the following rules concerning the use of the) option. The ANSYS commands that these rules apply to are those used with the **:C** and **Cmd_** block commands and do not use any blank spaces in the command.

- The suppressed command is *not likely* to be used as an abbreviation or macro name.
- The suppressed command is *not required* when the log file is run in batch. That is, the output from the batch run of the log file should not be different from that of an interactive run.
- The suppressed command *does not* affect any subsequent ANSYS commands that do not use the) option, and the result of the) command does not affect any commands issued after it.
- The result of the suppressed command nullifies *only* the effect of a previous ANSYS command that *did use* the) option.

5.2.1. General Guidelines for Command Suppression

- The ***GET** commands and other APDL commands that begin with an * (asterisk) used in preparing defaults for function data fields qualify for using this option.
- Do not suppress x**PLOT**, x**LIST**, and so on, commands even though these commands are not absolutely required for a batch run.
- You can suppress module name commands (NODE, ELEM, MESH, and so on), and a following **STATUS** command.
- Commands on the same line (using $) should individually use this option if they need to be suppressed. For example, in `:C )*GET,_z1,... $*DEL,_z1`, the ***DEL** command will be written to the log file. To suppress both commands, use:

```
:C )*GET,_z1....    $)*DEL,_z1
```

- Although ")" commands are not written to the command log, their action is echoed to the primary output when they execute. To avoid this output, enclose the suppressed commands with **/NOPR** and **/GO** as follows:

Suppress command echo.

```
:C )/NOPR
```

Following commands to be suppressed.

```
:C )*GET,_z1,....
:C )*GET,_z2,....
:C )_z3=_z1+_z2
:C )*DEL,_z1
```

Resume command echo.

```
:C )/GO
```

5.2.2. Examples of Command Suppression

Valid

Get a value from database.

```
:C  )*GET,_z1,......
```

Command nullifies previous *GET.

```
:C  )*DEL,_z1
```

Invalid

Fails Rule 3 (_z1 used subsequently).

```
:C  )*GET,_z1,......
:C  N,1,_z1,3,1
```

Invalid

```
:C  *GET,_z1,....
:C  N,1,_z1,3,1
```

Clean-up *GET value that did not use ")".

```
:C  )*DEL,_z1
```

5.3. Chained Function Blocks

Chained function blocks provide a mechanism for calling different commands or functions based on user input. They can also use hidden dialogs to process data.

The following example demonstrates chained function blocks. This example executes the **/CLEAR** command, which requires that the user be at the Begin level of the program. Chaining determines the current level of the program and prompts to execute the **/CLEAR** command.

Chained function blocks should not contain an **Apply** button until the last dialog in the chain (and then only if one is needed in the last dialog box).

```
:N Fnc_/CLEAR
:S      0,     0,     0
:T Cmd
:A Hid
:C )! Fnc_/CLEAR
:C )/NOPR
```

Set the chain return (not required for most chains).

```
:C )_z9 = '/CLEAR_2'
:C )KEYW,NOSHOW,1
:C )/GOPR
Inp_P
Cmd_)!
K_LN(begin)
```

If not at the begin level check which routine ANSYS is in.

```
Cal_Fnc_CHK_ROUT
K_LN(BEGIN)
```

At the begin level prompt to clear the database.

```
Cal_Fnc_%_z9%
:E END
:!
:N Fnc_/CLEAR_2
:S      0,     0,     0
:T Cmd
:C )! Fnc_/CLEAR_2
:D Clear Database and Start New
:H Hlp_C_/CLEAR
Inp_Noapply
 Fld_0
 K_FL(begin)
  Typ_Lab
  Prm_[/CLEAR]  Clear database and start new!
```

```
 Fld_0
 K_FL(BEGIN)
  Typ_Lab
  Prm_[/CLEAR]  Clear database and start new!
Cmd_FINISH
 K_CM(begin)
 Fld_0
  Typ_Lab
  Prm_                       *** WARNING ***
 Fld_0
  Typ_Lab
  Prm_This function requires that files currently open be
closed,
 Fld_0
  Typ_Lab
  Prm_continuing will exit
%_z2%%_z3%%_z4%%_z5%%_z6%%_z7%.
 Fld_0
  Typ_Sep
Cmd_/CLEAR
 Fld_2
  Prm_Read start.ans after clear?
  Typ_Lis_RadioB
   Lis_Read file       ,START
   Lis_Do not read file,NOSTART
:E END

:N Fnc_CHK_ROUT
:S      0,     0,     0
:T Cmd
:C )! Fnc_CHK_ROUT
```

When creating a function, add some explanation using the comment control.

```
:! EXPLANATION:
:! This function block is used to determine the current
:! ANSYS routine, and is meant to be only called by another
:! function block
:!
:! Argument _z9 must be passed to this function block for
:! the return of control for the function blocks.  The keyword
:! NOSHOW can be set true to cause this to be a hidden function.
:!
:C )/nopr
:C )_z2='the curr' $)_z3='ent rout' $)_z4='ine' $)_z5='' $)_z6='' $)_z7=''
:C )*GET,_z1,active,,routin
:C )*IF,_z1,EQ,17,THEN
:C )  _z2='the prep' $)_z3='rocessor' $)_z4=''
:C )  _z5='' $)_z6='' $)_z7=''
:C )*ELSEIF,_z1,EQ,21,THEN
:C )  _z2='solution' $)_z3='' $)_z4=''
:C )  _z5='' $)_z6='' $)_z7=''
:C )*ELSEIF,_z1,EQ,31,THEN
:C )  _z2='the gene' $)_z3='ral post' $)_z4='process'
:C )  _z5='or' $)_z6='' $)_z7=''
:C )*ELSEIF,_z1,EQ,36,THEN
:C )  _z2='the time' $)_z3='-history' $)_z4=' postpro'
:C )  _z5='cessor' $)_z6='' $)_z7=''
:C )*ELSEIF,_z1,EQ,41,THEN
:C )  _z2='optimiza' $)_z3='tion' $)_z4=''
:C )  _z5='' $)_z6='' $)_z7='        '
:C )*ELSEIF,_z1,EQ,52,THEN
:C )  _z2='the bina' $)_z3='ry file ' $)_z4='dumping '
:C )  _z5=' process' $)_z6='or' $)_z7='        '
:C )*ELSEIF,_z1,EQ,62,THEN
:C )  _z2='the radi' $)_z3='ation ma' $)_z4='trix gen'
:C )  _z5='eration ' $)_z6=' process' $)_z7='or      '
:C )*ELSEIF,_z1,EQ,65,THEN
:C )  _z2='the IGES' $)_z3=' file tr' $)_z4='ansfer p'
:C )  _z5='rocessor' $)_z6='        ' $)_z7='        '
:C )*ELSEIF,_z1,EQ,71,THEN
:C )  _z2='the run ' $)_z3='time sta' $)_z4='tistics '
```

```
:C )  _z5=' process' $)_z6='or' $)_z7=''
:C )*ENDIF
:C )/GO
:D                            WARNING
:H Hlp_C_FINISH
```

By using line filtering this function is made to be hidden or a dialog box.

```
K_LN(BEGIN*NOSHOW)
Inp_P
K_LN(begin+noshow)
Fmt_H
K_LN(begin+noshow)
Inp_NoApply
Cmd_FINI
K_CM(begin+noshow)

 Fld_0
  Typ_Lab
  Prm_                        *** WARNING ***
 Fld_0
  Typ_Lab
  Prm_This function requires that files currently open be closed,
 Fld_0
  Typ_Lab
  Prm_continuing will exit %_z2%%_z3%%_z4%%_z5%%_z6%%_z7%.
 Fld_0
  Typ_Sep
```

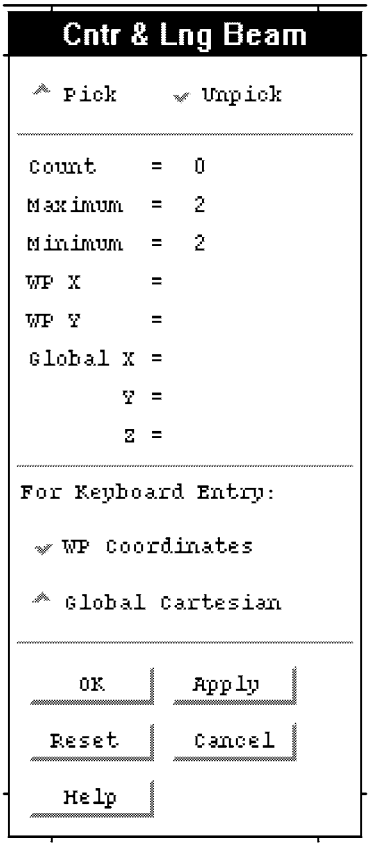
The keyword used by the chain is set back to zero and the UIDL processing is passed back to the _z9 parameter.

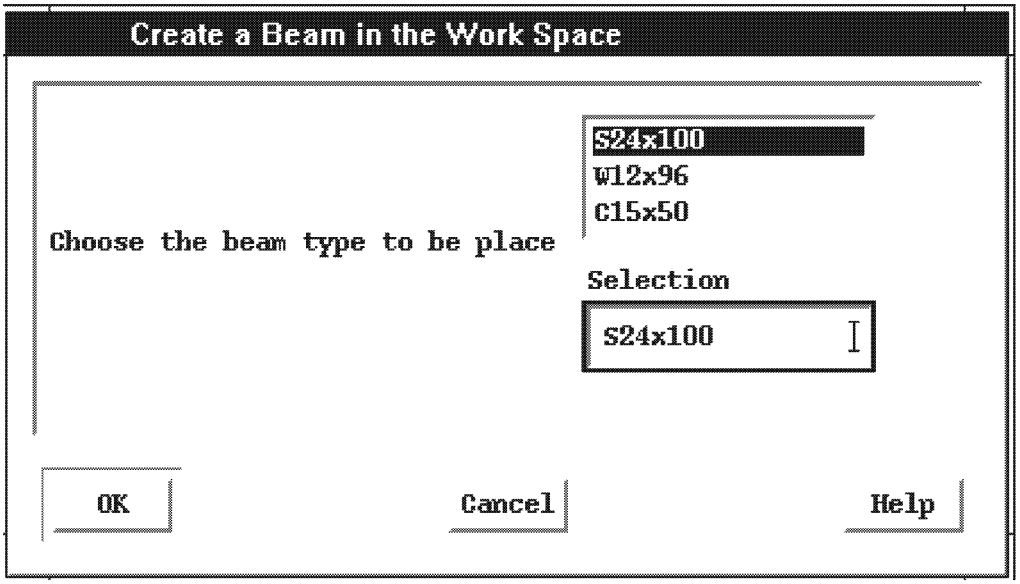
```
Cmd_)KEYW,NOSHOW,0
Cal_Fnc_%_z9%
:E END
```

Chapter 6: Programming Example

The following function block, PlaceBeam, uses a picking box to store the picked keypoints, then calls PlaceBeam2 to create a beam. PlaceBeam uses locational picking, and the **FLST** and **FITEM** commands store the picked coordinates. PlaceBeam2 uses the ***FPIK** control to get the location for each field.

```
:N Fnc_PlaceBeam
:S      0,     0,     0
:T Cmd_P
:A Custom Part
:C )! Fnc_CustomPart
:D Cntr & Lng Beam
:H Hlp_C_PlaceBeam
Inp_P
Cmd_!
 Fld_2
  Prm_[PlaceBeam] Pick WP location
 or enter coordinates
  Typ_XYZ
  Rub_1
  Min_2
  Cnt_2
  Pfm_1
  Pwr_1
Cal_Fnc_PlaceBeam2
:E End
:!
:N Fnc_PlaceBeam2
:S      0,     0,     0
:T Cmd_P
:A On Working Plane
```






```
:D Create a Beam in the Work Space
Inp_NoApply
Cmd_PlaceBeam
  Fld_2
```

```
      Prm_Choose the beam type
      to be placed.
      Typ_Lis
       Lis_S24x100,0
       Lis_W12x96 ,1
       Lis_C15x50 ,2
    Fld_3
       Typ_Def_*FPIK(2,1)
    Fld_4
       Typ_Def_*FPIK(2,2)
    Fld_5
       Typ_Def_*FPIK(2,3)
    Fld_6
       Typ_Def_*FPIK(2,4)
    Fld_7
       Typ_Def_*FPIK(2,5)
    Fld_8
       Typ_Def_*FPIK(2,6)
  :E End
```

Appendix A. UIDL Command Dictionary

UIDL Commands and Controls

This chapter describes the commands and controls for GUI programming. The descriptions include examples of command or control usage, except in cases where you specify only the command or control with no additional arguments.

:! *String*
Indicates a comment line or block separator.

`Header`

Argument Description

`String`
A string containing 1 to 76 characters.

Notes

Comment lines are recommended for separating building blocks. Insert a **:!** command at the beginning of each new block header.

Avoid using a **:!** on a line by itself--insert a blank space after the **:!** to create a block separator without a comment.

! *String*
Indicates a comment line.

`Data Control`

Argument Description

`String`
A string containing 1 to 76 characters.

Notes

Avoid using a **!** on a line by itself--insert a blank space after the **!**.

:A *String*

Defines the string to display on the menu calling this building block. (Required)

`Menu Header,Function Header`

Argument Description

`String`

A 1- to 16-character string which names the menu. If this is a menu building block, a greater-than symbol > will be appended to the string in column 18.

Notes

The name of this building block can match the name of other building blocks.

This string is not the same as the internal name defined by the **:N** command. The string defined by the **:N** command is the internal name of the menu; it is only used for programming and is not displayed to the user.

Bnd_*LOWER,UPPER*

Provides an upper and lower limit for the Typ_MLis control.

`Function Data Control`

Argument Description

`LOWER,UPPER`

A lower and upper limit for the multiple list. Both limit values must be double precision numbers. UPPER cannot be greater than 10.

:C *Command*

Specifies a command to be executed upon entry to a building block.

`Header`

Argument Description

`Command`

Any valid 1- to-76-character command string. The string can consist of a standard ANSYS command, a line of APDL code, a user-programmable feature or a call to an external libraries. See the *ANSYS APDL Programmer's Guide* for more information on APDL.

Notes

You can specify multiple **:C** lines inside a building block, but they must be in the Header section of the block and they should be listed together.

Cal_Fnc_Block,*FIELD*,*Oper*,*VALUE*,*CMDNUM*

Calls another function block, based on an if-test for a specified field.

Function Data Control

Argument Description

Fnc_Block
: The internal name of the function block to be called (as defined by a **:N** command) if the if-test is satisfied.

FIELD
: The number of the field to be evaluated.

Oper
: The comparison operation to use for the evaluation. Valid operations are: EQ, NE, LT, GT, LE, GE, and ES (string equality).

VALUE
: The value with which to evaluate *FIELD*.

CMDNUM
: The order of the command in this data block whose field the test will use.

Notes

Omit the *FIELD*,*Oper*,*VALUE*, and *CMDNUM* arguments for an unconditional callback of a function block. You may use parameter substitution by enclosing a parameter name with the % symbol. Use %*(I)name*% to force conversion of a numeric parameter to integer form.

When a call is encountered, it executes and the processing returns immediately to the next line after the call and continues processing. A dialog invoked by the first instance of **Cal**_ will be replaced by the dialog invoked by the second instance of **Cal**_. We do not recommend using this sequence to call a hidden dialog followed by a call to another dialog.

Cal_ should not be used to call a function block which is the current call stack (i.e. *recursive*). This could result in a stack overflow and a subsequent fatal execution error.

This command must be the last line in a function block, immediately preceding the **:E END** command.

Examples

The following command will call Fnc_*Block* if field number 4 of the second command (the second Cmd_*Block*) in the current data control has a value greater than 1 when the current function block executes:

```
Cal_Fnc_Block,4,GT,1,2
```

The command below calls Fnc_TESTCALL for parameter name='CALL':

```
Cal_Fnc_TEST%name%
```

Cal_REFRESH

Refreshes the lowest active level of the main menu.

`Function Data Control`

Argument Description

If input entered in a dialog effect the keywords used to activate a menu item, the menu is refreshed to reflect the new settings.

Cmd_*Command*

Constructs and issues an ANSYS command as part of a function data control block.

`Function Command Control`

Argument Description

`Command`

Any valid ANSYS command name. This includes APDL calls, macros, and calls to external libraries.

Notes

You can include more than one **Cmd_** data control in a single data block, but not more than 50. If this command shouldn't be included in menu paths, then on the preceding line, add this text:

```
! Exclude_from_paths
```

Example

The following command issues ANSYS command **APLOT**, which displays an area plot in the Graphics Window:

```
Cmd_APLOT
```

See Section 5.2: Suppressing ANSYS Commands for details on the use of) and $

Cnt_*INTEGER*

Defines the maximum number of items users can pick for this field, without automatically invoking Apply.

`Function Picking Data Control`

Argument Description

`INTEGER`

An integer greater than or equal to 1 but not higher than 1000 (for locational picking) and not more than 9999 (for entity picking). If you omit the integer, it defaults to 1. You can replace *`INTEGER`* with an "l" (the letter l) or an "s," which doesn't limit entity picking. The "l" is a flag indicating an ordered list of picked items. The "s" is a flag indicating an unordered list of picked items.

Notes

This command is useful when a user is picking items to create or select and the picked items should be verified before performing the next operation. See **Max_** to see how to set a maximum value and invoke Apply automatically. See **Mok_** to see how to set a maximum value with an automatic OK.

Examples

This command sets the order of selection as unimportant for "Select Entities:"

```
Cnt_s
```

This command limits the number of picked keypoints to 8:

```
Cnt_8
```

---

*CPAR(*N*)

Retrieves data stored in the *CSET command vector.

`Function GUI Control`

Argument Description

(*N*)
The number of the appropriate vector position. This command usually is used in conjunction with the **Typ_Def_** command.

Example

The following commands instruct the ANSYS program to use the data stored in the ***CSET** vectorposition 101 as the default value:

```
Typ_Def_*CPAR(101)
```

CPAR

Fetches the double-precision equivalent of the data stored in the *CSET command from an ANSYS *GET command.

`Function GUI Control`

Explanation

This command usually is used in conjunction with the **Typ_Def_** command.

Example

The following command assigns the double-precision equivalent of the data stored in the CSET vector position 101:

```
*GET,_z1,CPAR,101
```

***CSET**,*START_LOC*,*END_LOC*,*Fields*

Stores a string in a vector.

Function GUI Control

Argument Description

START_LOC

An integer between 1 and 150.

END_LOC

An integer between 1 and 150. To maintain precision, use the START_LOC + 3 for the ending location.

Fields

The string to be stored at the location starting at *START_LOC* and ending with *END_LOC*. Fill these fields using **Fld_** controls; each field can hold as many as 32 characters. If the string contains spaces, you must enclose the complete string in single quotation marks, as follows:

```
`my string'
```

Example

This command fills the ***CSET** vector positions 1 through 1 with the following field specifications

```
Cmd_*CSET,1,1
Fld_2:
  Prm_Name of the ANSYS job
   Typ_Char
Cmd_*CSET,1,1,`This is a test'
```

Notes

When settings defaults for filtering, use parameters to set the value first. For example, in an IGES import, you cannot set `*.*` or `*` in the **Def_** statement. Instead, a ***PAR** of a parameters such as `_Z10` is used `(_Z10="*.*")`.

!Cust_Cal_*Function_Name*

Names the function block which the custom dialog calls.

Function Data Control

Notes

This command can only appear after a comment command (**!**).

!Cust_Cmd_*Command_Name*

Names the ANSYS command which the custom dialog calls.

Function Data Control

Notes

This command can only appear after a comment command (**!**).

:D *String*

Defines the title of the dialog, and the description of a control file or a menu. (Required for control file and function block headers.)

`Control File Header,Function Header,Menu Header`

Argument Description

`String`
A string of 1 to 76 characters.

Notes

Only the first 60 characters of the string appear in the dialog's title bar. Only the first 22 characters of the string will show in a picking dialog. If you reset the default font size for the dialog boxes, a different number of characters will appear.

In menu blocks, this command can be used to store descriptive information about the menu, but the information is not displayed to the user.

Def_*String*

Defines a default value to be used and displayed in this field.

`Function Field Control`

Argument Description

`String`
The *`String`*, limited to eight characters in length,may be any of the following:

- The string Blank, which will produce a blank input field. The field also defaults to a blank input field if the value returned from ***GET**, ***PAR**, etc. is tiny (2**-100).
- A numeric value, which sets the default. If the user changes the value for this field, then it overrides the setting for *`String`*. The user will see the changed value if no more than 40 other dialog boxes have been shown before this same dialog is called again.
- A ***GET** value or a ***PAR** value which overrides previous user values. More properly, though, the ***GET** value should match the user value so that when the command is executed, the database is updated and reflects the new user value. Thus, a subsequent entry will get the user value from the database.
- A ***CPAR** value which overrides previous user values.
- A ***PICK** value, which uses the information from the **Typ_XYZ_WP** function picking control.
- A ***FENT** or ***FPIK** value.
- A ***Str** value, which passes up to 72 characters into a field as one character.
- A string matching an item in a **Lis**_ data control. If the string contains spaces, enclose it in single quotes.

You can specify up to three default values, separated by commas, as the defaults for double and triple data types (for example, **Typ_Real2**, **Typ_Real3**).

Examples

Def_*GET (*Entity*,*ENTNUM*,*Item1*, *IT1NUM*,*Item2*,*IT2NUM*)	Gets a value the same way as the ANSYS ***GET** command
Def_*PAR(*Parm*)	Evaluates *Parm*, which is an ANSYS parameter
Def_*PICK(*N*)	Gets the field *N* information from the **Typ_XYZ_WP** picker, where *N* is a number from 1 to 9.
Def_*PAR(_z1),***PICK**(3),5	Used for a triple data type
Def_*CPAR(101)	Use the 101 value in the CSET vector
Def_Blank	Produces a blank input field

Dlm_*Character*

Defines the command field delimiter the UIDL processor will use when building the command to send to ANSYS.

Function Field Control

Argument Description

Character

A character, which can be a blank space, to use *after* this field as the delimiter. The default character is a comma.

The character "~" causes ANSYS to use no delimiter between fields. Fields are compressed to the last non-blank character.

Example

The following data block invokes the vi editor for the given filename in the ANSYS Output Window, and sets the delimiter character to a space:

```
Cmd_/SYS
  Fld_2

  Typ_Def_vi
  Dlm_                Note--a single space follows Dlm_

  Fld_3
  Typ_CHAR
  Prm_Enter Filename
```

:E END

Ends every building block. (Required)

Ending

Notes

Most blocks end with only the **:E END** command in the ending section. However, when you are creating online help, you need to insert a blank line immediately before the **:E END** command.

:F *String*

Defines the name of the GUI control file. (Required)

Control File Header

Argument Description

String
The name of this control file.

Example

The following example shows the first four required lines of a control file:

```
:F UIFUNC.GRN
:D Modified on %E%, defines standard ANSYS dialogs
:I      0,       0,       0
:!
```

---

*FENT(*FIELD,ITEM*)

Retrieves data stored by the FLST and FITEM commands for entity picks.

Function Data Control

Argument Description

FIELD
The field number associated with the ANSYS command used for the picking. This can be a value between 2 and 9.

ITEM
The item number for the entities picked by *FIELD*. The item number can be between 0 and 999. A zero value for the item number returns the number of entities picked.

Explanation

***FENT** builds commands based on data previously stored by the **FLST** and **FITEM** commands from a previous function block. It is typically used in conjunction with the **Typ_Def_** command.

Fld_*N*

Specifies the field number

`Function Field Control`

Argument Description

N

The field number of the ANSYS command field. The field definition always begins with **Fld_**, and ends with the next **Fld_** or **Cmd_** control. The **Cmd_** control immediately preceding the **Fld_** is related to the field.

Notes

The maximum number of **Fld_** data controls allowed in a function is 80.

Specify **Fld**_0 for labels. These will appear in a dialog box as a separate text line, without a text input box attached. **Fld**_1 should not be used explicitly; UIDL associates **Fld**_1 with the command used for the dialog.

Example

The following example defines **Fld**_2 as a labeled field that will accept a character string of up to eight characters in length:

```
Fld_2
Prm_Name of parameter to be defined
Typ_Char,8
```

Fmt_H

Forces narrow horizontal sizing of dialogs.

`Function Data Control`

Notes

Do not use this command with the **Inp_P** control. Any **Prm_** labels will be shown above their corresponding button or data entry field.

Fnc_*String*

Points to the name of a function block from a menu, or defines the internal name of a function. (Required in function headers.)

`Menu Data Control`

Argument Description

String

The internal name of the function block, indicating a function to be displayed or activated when the user chooses this item from a menu.

Notes

The contents of *String* are defined by a **:N** command in a function header.

---

*FPIK(*FIELD,ITEM*)

Retrieves data stored by the FLST and FITEM commands for locational picks.

Function Data Control

Argument Description

FIELD

The field number associated with the ANSYS command used for the picking. This can be a value between 2 and 9.

ITEM

The item number associated with the X, Y, or Z coordinate (in the global Cartesian coordinate system) of the entity chosen with Field. The item numbers correspond to the coordinates of the entities that were picked. The item number can be between 0 and 999 (i.e., there can be 333 coordinate locations). A zero value for the item number returns the number of points picked. Always double check the return value.

Explanation

***FPIK** builds commands based on data previously stored by the **FLST** and **FITEM** commands from a previous function block. It is typically used in conjunction with the **Typ_Def_** command.

Example

See Chapter 6, "Programming Example" for a programming example using ***FPIK**.

GET(String)*

Gets a value without first placing it in a parameter.

Function GUI Command

Argument Description

String

The same format as the ANSYS ***GET** command after the parameter field.

Entity

Entity keyword. Valid keywords are NODE, ELEM, KP, LINE, AREA, VOLU, etc; see *Entity*= in the ***GET** table in the *ANSYS APDL Programmer's Guide*.

ENTNUM

The number of the entity (as shown for *ENTNUM*= in the ***GET** table in the *ANSYS APDL Programmer's Guide*). 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 ***GET** table in the *ANSYS APDL Programmer's Guide*.

IT1NUM
The number (or label) for the specified *Item1* (if any). Valid *IT1NUM* values are as shown in the *IT1NUM* columns of the ***GET** table in the *ANSYS APDL Programmer's Guide*. 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

This command is useful for working with **Def_** data controls. See the *ANSYS APDL Programmer's Guide* for detailed information on the ***GET** command. For ***GET** operations on character commons, CHR4 reads through the common in blocks of four and CHR8 reads through the common in blocks of eight.

:H Hlp_*String*
References a particular help file.

Function Header

Argument Description

String
The name of the help that should be displayed when a user clicks the Help button of the dialog box.

Example

The following command displays the online help for the **RECTNG** command when a user looks for information about that command:

```
:H Hlp_C_RECTNG
```

Notes

To be able to readily create consistent names for help blocks, ANSYS has adopted the following naming conventions:

- **Hlp_***String* for a user-created help.
- **Hlp_C_***ANSYSCommandName* for a command documented in the *ANSYS Commands Reference* (the internal name for the help name for **AADD** is **Hlp_C_AADD**). A user can enter **Help,** ANSYSCommandName in the ANSYS Input Window to display online help about the specified command.
- **Hlp_E_***ELEMENTNAMENUMBER* for an element documented in the *ANSYS Elements Reference* (the internal name for the help name for LINK8 is **Hlp_E_LINK8**). A user can enter **Help,** *ELEMENTNAMENUMBER* in the ANSYS Input Window to display online help about the specified element.
- **Hlp_G_***String* for the *Analysis Guides*, where String is provided by the Documentation Group (Hlp_gask for the Gasket Materials section in the *ANSYS Elements Reference*.
- **Hlp_UI_***String* for a screen in the *ANSYS Online Help Topics*.

When you are creating help names for function blocks that you've written, you should give the help a name similar to the function block to ease software maintenance.

:I 0, 0, 0

Fixed-format placeholder for the UIDL processor-generated index numbers. (Required)

`Control File Header`

Explanation

The **:I** command must appear as the third line of the control file header, with comma-separated zeros in columns 9, 18, and 27 respectively.

When the UIDL processor processes this file, the placeholders are overwritten with index numbers. Always keep a clean copy of every control file with the index line set to 0s.

The first field is a placeholder for the total number of building blocks. The second field is the placeholder for the starting location of the **:X** INDEX in bytes. The third field is a placeholder for the total size of the control file in bytes.

Inp_NoApply

Suppresses the Apply button in a dialog.

`Function Data Control`

Explanation

When **Inp_NoApply** is used, it must be the first entry in the data controls section. Do not use **Inp_NoApply** with the **Inp_P** control.

Inp_NoApply must be used with unique specification commands (not available for multiple windows, sets, etc.), an action performed only once, and all visible dialogs of a chain except the last visible one (if an **Apply** is appropriate)

Inp_P

Acts as a prompt mode flag for including a picking dialog or for a hidden block.

`Function Data Control`

Notes

This command, if used, must be the first entry in the function data controls block. *Don't use this control if a dialog box is to appear for the function block.* Do not use this control with the **Inp_NoApply** command.

If you use **Inp_P** with a data type (**Typ**_*Control*) that includes picking, the picking dialog will be displayed in addition to the prompt window. *Do not* use this command to display a prompt window without a picking dialog.

:K*Keyword Logic*

Defines logic that must produce a true result before a dialog is shown or menu selections are activated.

Header

Argument Description

Keyword Logic

Keyword Logic is an algebraic Boolean operation--"|" is **or** and "&" is **and**. You can enclose the logic in parentheses nested up to three levels deep. An uppercase keyword requires that the keyword be set for a true condition to be returned. A lowercase keyword requires that the keyword *not* be set for a true condition to be returned. If the keyword produces a false condition, the block is *not* displayed. If you place a "#" symbol before the first parenthesis, the menu item corresponding to the building block is grayed out instead of not being displayed.The *Keyword Logic* string can contain up to 76 characters. Keywords may be selected by ANSYS or you can include them in your programming.

Notes

You can include more than one **:K** line in the Header Section but not more than 10. When you use multiple **:K** lines, the ANSYS program uses the **and** logic between them. This enables you to dim or hide a function based on keywords.

Appendix B:, ANSYS Keywords lists and describes the keywords ANSYS uses.

Filtering only works in the first level of the Utility Menu--filtering does not work in cascaded menus.

Example

The following command specifies that either "PREP7" **or** "SOLUTION" must be **true**, **and** "BEGIN" must be **false** in order for the GUI to display the block--ANSYS must be running either in preprocessor or solution mode.

```
:K ((PREP7|SOLUTION)&begin)
```

K_*Type*(*Keyword Logic*)

Defines keyword logic that must produce a true result before the identified item will be shown.

Function Data Control

Argument Description

Type

The type of keyword logic used. Possible values are:

CM (applies keyword logic to the previous **Cmd**_)
FL (applies keyword logic to the previous **Fld**_)
LN (applies keyword logic to the following line, no matter what control type it is. You can use an unlimited number of **K**_LN lines.)

Keyword Logic

The *Keyword Logic* is an algebraic Boolean operation, where "|" is **or** and "&" is **and**. You can enclose the logic in parentheses nested up to 3 levels deep.<"argexplain">An uppercase keyword requires that the keyword be set for a true condition to be returned. A lowercase keyword requires that the keyword *not* be

set for a true condition to be returned. If the keyword produces a false condition, the block *is not* displayed. If you place a "#" symbol before the first parenthesis, the menu item corresponding to the block is dimmed (grayed out) instead of not being displayed. The *Keyword Logic* string can contain up to 76 characters. Keywords may be selected by ANSYS or you can include them in your programming.

Notes

You can include more than one **K**_LN line where **and** logic is used between lines. Appendix B:, ANSYS Keywords lists and describes the keywords ANSYS uses.

The **K**_LN is not to be used with P_LN in a menu block. If the combination is needed use the **:K** and **:P** lines of the header section of the function block or menu being called.

Max_*INTEGER*

Sets the maximum number of items users can pick for this field, automatically invoking Apply.

Function Picking Control

Argument Description

INTEGER

An integer greater than or equal to 1, but not exceeding 1000 (for locational picking) or 9999 (for entity picking). If you omit the integer, it defaults to 1.

You can replace the integer value with an "I" (the letter I) or an "s," which doesn't limit entity picking. The "I" is a flag indicating an ordered list of picked items. The "s" is a flag indicating an unordered list of picked items.

Notes

The ANSYS program does an "Apply" automatically when the user picks the maximum number. See the **Cnt**_ description for information on how to specify a maximum value without doing an Apply. See the description of **Mok**_ to find out how to specify a maximum value with an automatic OK.

Examples

The following command defines the order of selection as unimportant:

```
Max_s
```

The following command sets the maximum number of picked items at eight:

```
Max_8
```

Men_*String*

Points to the name of a menu block to display on a menu.

`Menu Data Control`

Argument Description

`String`

The internal name of the menu block, which indicates the menu to be displayed when the user picks this item in a menu.

Notes

The contents of *`String`* are defined by a **:N** command in a menu header.

Min_*INTEGER*

Sets the minimum number of items that users must pick for this field.

`Function Picking Control`

Argument Description

INTEGER

An integer greater than or equal to 1. If you omit the integer, the default is 1; however, a good practice is to specify a minimum integer value and not rely on the default.

Notes

This control sets the minimum number of items that must be picked before the ANSYS GUI will accept an Apply or an OK.

Example

The following control sets the minimum number of picked items to four:

```
Min_4
```

Mok_*INTEGER*

Sets the maximum number of items users can pick for this field.

`Function Picking Control`

Argument Description

`INTEGER`

An integer greater than or equal to 1, but not exceeding 1000 (for locational picking) or 9999 (for entity picking). If you omit the integer, it defaults to 1.

You can replace the integer value with an "l" (the letter "l") or an "s," which doesn't limit entity picking. The "l" is a flag indicating an ordered list of picked items. The "s" is a flag indicating an unordered list of picked items.

Notes

The ANSYS program does an OK operation automatically when the maximum is picked. See the **Cnt_** description for information on how to specify a maximum value without an automatic Apply. See the description of **Max_** to find out how to specify a maximum value with an automatic Apply.

:N *String*

Defines the internal name of a building block. (Required)

`Header`

Argument Description

`String`

A unique 1- to 12-character name, prefixed with one of the following:

Fnc_, for a function block
Men_, for a menu block

Notes

If the same internal block name is defined more than once, ANSYS uses only the *last* building block found with that name. Each internal name should be unique. See the **:H** description for more information on naming help blocks.

:P *Product Code Logic*

Defines the product codes that must be present before the building block will be shown.

`Header`

Argument Description

`Product Code Logic`

The ANSYS program sets product codes. (A product code identifies the ANSYS products for which your site is licensed.) The product code logic is an algebraic Boolean operation where "|" is **or** and "&" is **and**. You can enclose the product code logic in parentheses nested up to 3 levels deep.

An uppercase product code requires that the product code be set for a true condition to be returned. A lowercase product code requires that the product code *not* be set for a true condition to be returned. If the product code produces a false condition, the building block *will not* be displayed. If you place a "#" symbol before the first parenthesis, the menu item corresponding to the building block will be dimmed (grayed out) instead of not being displayed. The *`Product Code Logic`* string can contain up to 76 characters.

Notes

You can use only one **:P** command in the header section of a function block.

Appendix C:, ANSYS Product Codes of this manual lists and describes the ANSYS product codes.

Filtering only works in the first level of the Utility Menu--filtering does not work in cascaded menus.

Example

The following command requires the THERMAL product code to be active before the building block will be displayed (that is, a product with thermal capabilities (i.e., ANSYS Mechanical) must be running on the system):

```
:P (THERMAL)
```

P_*Type* (*Product Code Logic*)

Defines product code logic that must produce a true result before the identified item will be shown.

`Function Data Control,Menu Data Control`

Argument Description

`Type`

The type of product code logic to use. Specify one of the following:

CM (product code logic applies to the previous **Cmd**_)
FL (product code logic applies to the previous **Fld**_)
LN (product code logic applies to the following command, no matter what control type it is)

`Product Code Logic`

Product code logic is an algebraic Boolean operation, "|" is **or** and "&" is **and**. You can enclose the logic in parentheses nested up to three levels deep.<"argexplain">An uppercase product code requires that the keyword be set for a true condition to be returned. A lowercase product code requires that the keyword *not* be set for a true condition to be returned. If the keyword produces a false condition, the building block *will not* be displayed. If you place a "#" symbol before the first parenthesis, the menu item corresponding to the building block will be dimmed (grayed out) instead of not being displayed.The *`Product Code Logic`* string can contain up to 76 characters.

A product code in uppercase letters required that the product code be set to "true." A lowercase product code must be set to false. If a product code produces a false condition, the building block *isn't* displayed.

The *`Product Code Logic`* string can contain up to 76 characters.

Notes

Appendix C:, ANSYS Product Codes of this manual lists the ANSYS product codes.

The **P**_LN is not to be used with **K**_LN in a menu block. If the combination is needed use the **:P** and **:K** lines of the header section of the function block or menu being called.

---

*PAR(_ZString)

Retrieves data stored in a parameter.

Function GUI Control

Argument Description

_ZString

A parameter that was previously defined. This command is used in conjunction with the **Def_** or **Typ_Def_** command.

Example

The following example uses the parameter _Z1 as the default node number.

```
Fld_4
Prm_Node number N
Typ_INT
Def_*PAR(_Z1)
```

Pcn_*FLAG*

Specifies whether a set of picked entities should or should not be ordered for retrieval picking.

Function Picking Control

Argument Description

FLAG

One of the following values:

0, for an ordered set (default)
1, for an unordered set
2, where the first item chosen is the first in the set and the other items are unordered. This is available for loop picking and single picks.

Notes

For an ordered set, the following picking dialog buttons will be dimmed: Box, Polygon, Circle, and Pick All.

Pdp_*FLAG*

Controls whether the user can pick duplicates of an entity for retrieval picking

Function Picking Control

Argument Description

FLAG

One of the following values:

0, if no duplicates are allowed (default)
1, to allow duplicates of an entity

Pfm_*FLAG*

Controls how the ANSYS program processes picked items.

`Function Picking Control`

Argument Description

FLAG

One of the following values:

0, to have the UIDL processor determine if a picked set or the command line should be used (default).
1, to use a picked set (The ANSYS commands **FITEM** and **FLST** will be used.)
2, to place the picked items directly on the command line (The ANSYS commands **FITEM** and **FLST** will *not* be used.)

***PICK**(*N*)

Retrieves the data in the *N*th text field of the model picker.

`Function Picking Control`

Argument Description

(*N*)

The text field number of the model picker

Example

In the following example, two working plane locations (the two diagonal corners of a rectangle) are picked:

```
:N Fnc_BLC4_2d
:S      0,     0,     0
:T Cmd_P
:c )! Fnc_BLC4_2d
:A By 2 Corners
:D Rectangle by 2 corners
:K #(PREP7)
:H Hlp_C_BLC4
Inp_P
Cmd_BLC4
 Fld_0
    Typ_XYZ_WP
    Prm_[BLC4] Pick 2 WP locations (2 diagonal corners)
    Min_2
    Max_2
    Rub_
 Fld_2
    Typ_Def_*PICK(1)

 Fld_3
  Typ_Def_*PICK(2)
 Fld_4
    Typ_Def_*PICK(3)
 Fld_5
    Typ_Def_*PICK(4)
 :E END
```

Prm_*String*

Defines the label (or prompt) to be used with this field.

`Function Field Control`

Argument Description

`String`

The string can contain up to 72 characters but will be truncated to 32 characters when in a dialog box for any data type (**Typ_**) other than **Typ_Lab**.

Notes

You can substitute ANSYS parameters in the labels by enclosing them with `%` symbols. Consider this command:

```
Prm_Specify %num% widgets that are %color%.
```

If the **`%num%`** parameter is set to 5 and the **`%color%`** parameter is set to green, the text string will read:

```
Specify 5 widgets that are green.
```

Use the **`%#parameter%`** to prevent compression of spaces in a character parameter.

Example

The following command specifies the prompt, "Enter fillet radius."

```
Prm_Enter fillet radius
```

Pwr_*FLAG*

Switches the FLST and FITEM off and on for a field.

`Function Field Control`

Argument Description

`FLAG`

One of the following:

0, to have **FITEM** and **FLST** write the items (default)
1, to disable writing of items by **FITEM** and **FLST**

Notes

This command enables the storage of picked items or coordinates for retrieval by the ***FENT** or ***FPIK** commands.

Rmk_

Rebuilds the dialog box after an Apply operation.

`Function Data Control`

Argument Description

Notes

This control must be the first line of the Data Controls section.

Rub_*N*

Specifies the type of rubber banding to be used in a picking operation.

`Function Picking Control`

Argument Description

N

A numeric key from the following list:

For 2-D Picking on the Working Plane	
1	Lines
2	Rectangle (Corner - Opposite Corner)
3	Circle (Center - Radius)
4	Annulus (Center - R1 & R2)
5	Partial Annulus
6	Rectangle (Center - Corner)
7	Circle (On Circle - On Circle)
8	Rectangle (Center - Edge)
13	3 Equal-sided Polygon
14	4 Equal-sided Polygon
15	5 Equal-sided Polygon
16	6 Equal-sided Polygon
17	7 Equal-sided Polygon
18	8 Equal-sided Polygon
For 3-D Picking Between Keypoints	
51	Straight Lines
52	Lines In Active Csys
53	Area In Active Csys
54	Area With Straight Lines
For 3-D Picking on the Working Place Plus Z Dimension	
101	Polyhedron
102	Block

103	Cylinder-solid
104	Cylinder-hollow
105	Partial Cylinder
106	Block (Center - Corner)
107	Cylinder-solid (On Circle - On Circle)
109	Cone
113	3 Equal-sided Polyhedron
114	4 Equal-sided Polyhedron
115	5 Equal-sided Polyhedron
116	6 Equal-sided Polyhedron
117	7 Equal-sided Polyhedron
118	8 Equal-sided Polyhedron
For Electrical Circuits (3 Points to Create)	
201	Inductor
202	Capacitor
203	Resistor
204	Circuit 1
205	Circuit 2
206	Circuit 3
207	Circuit 4

:S 0, 0, 0

Fixed-format placeholder for the UIDL processor-generated index numbers. (Required)

`Header`

Explanation

The **:S** command must appear as the second line of every header section, with comma-separated zeros in columns 9, 16, and 23 respectively.

When the UIDL processor processes a block, the placeholders are overwritten with index numbers. Always keep a clean copy of every block with the index line set to 0s.

The first field is a placeholder for the size of the block in bytes. The second field is the placeholder for the start index of the data controls section in bytes. The third field is a placeholder for the total size of the data controls section in bytes.

Sel_*FLAG*

Defines the type of selection picking that is allowed.

`Function Picking Control`

Argument Description

FLAG

One of the following values:

s, to allow only currently selected entities to be picked (default).
1, to pick all defined entities, whether or not they're selected. *Use this setting only for retrieval picking.*
u, to allow picking only entities that currently aren't selected.

Sep_

Displays a separator bar in a menu.

`Menu Data Control`

Argument Description

Need text or example to make this refentry valid.

***Str**(*String*)

Passes character information into a field as a character.

`GUI command`

Argument Description

(*String*)

A string containing up to 72 characters. You can include ANSYS parameters in the string by enclosing them with % symbols (%*parameter*%). Use %#*parameter*% to prevent compression of spaces in a character parameter. If a parameter causes the string to exceed 72 characters, the string is truncated. This command is used with the **Typ_Def_** and **Def_** commands.

Example

The following block reads a file into ANSYS:

```
F1=`/usr/loc`
F2=`al/ansys`
F3=`file.in`
F4=`p`

Cmd_/INPUT
   Fld_2
       Typ_Def_*Str(%F1%%F2%%F3%%F4%)
```

:T*Type*

Specifies the type of data controls a building block will use. (Required)

`Header`

Argument Description

`Type`

One of the following types:

Menu --
: Defines the block as a menu block. The data controls section contains the internal names of the submenus and functions. The ANSYS program automatically appends a ">" symbol on the menu to the names (specified on the **:A** line) of menu blocks having submenus.

Cmd --
: Defines the block as a function block. An ellipses (...) appears on the menu after the name of each dialog.

Cmd_N --
: Defines the block as a function block. No symbol will be shown (this indicates immediate action). This is used for a hidden dialog.

Cmd_P --
: Defines the block as a function block for a picking dialog. The symbol + appears on the menu after the name of each picking box.

Finish --
: Defines the block as a "finish block." Using this data type and the **FINISH** command collapses the side menus automatically to the top level when a user chooses this command.

Notes

Changing the function block type changes only the symbol shown, not the functionality of the function itself.

Typ_*Entity*

Uses picking mode to obtain entity numbers for commands requiring them.

`Function Picking Control`

Argument Description

`Entity`

Use one of the following values for *`Entity`*:

NODE	Pick node numbers from the screen.
ELEM	Pick element numbers from the screen.
KEYP	Pick keypoint numbers from the screen.
LINE	Pick line numbers from the screen.
AREA	Pick area numbers from the screen.
VOLU	Pick volume numbers from the screen.
TRAC	Pick trace point numbers from the screen.

Notes

Use this data type only in the **Inp_P** control mode.

Typ_Char,*FIELD1*,*FIELD2*

Creates a text entry field for character data display.

`Function Character Control`

Argument Description

FIELD1

An integer between 1 and 80 for maximum character input (defaults to 80).

FIELD2

An integer between 1 and 80 for the input field view width (defaults to 32).

Typ_Color

Creates an option button with 16 color choices available.

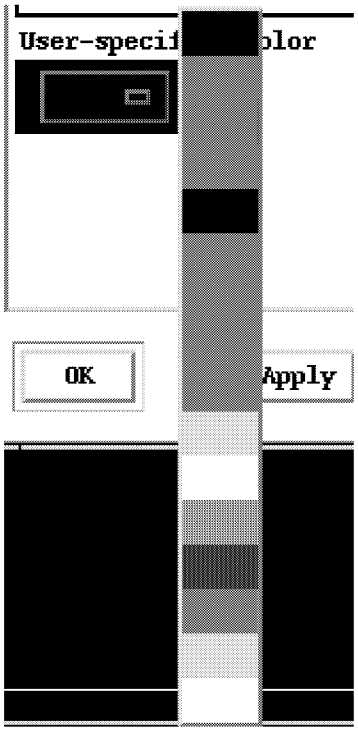
`Function Listing Control`

Explanation

The default prompt for **Typ_Color** is *Color*. The color chosen passes the actual color name to the field. If you want to set a default color (**Def_** line), the correspondence shown below applies for the default ANSYS color map. (If using a ***GET** command to fetch a color value, you must add 1 to it to match this table.)

1	BLAC	5	BLUE	9	GREE	13	RED
2	MRED	6	CBLU	10	YGRE	14	DGRA
3	MAGE	7	CYAN	11	YELL	15	LGRA
4	BMAG	8	GCYA	12	ORAN	16	WHIT

If you do not specify any parameters, ANSYS produces a selection list in the dialog that looks like this:



The user selects a color, and the button is shown in the color selected.

Typ_Def_*String*

Defines a hidden default value to be used but not displayed.

`Function Field Control`

Argument Description

`String`

The *`String`*, limited to eight characters in length, may be any of the following:

- A numeric value, which sets the default.
- A ***GET** value or a ***PAR** value which overrides previous user values. More properly, though, the ***GET** value should match the user value so that when the command is executed, the database is updated and reflects the new user value. Thus, a subsequent entry will get the user value from the database.
- A ***CPAR** value which overrides previous user values.
- A ***PICK** value, which uses the information from the **Typ_XYZ_WP** function picking control.
- A ***FENT** or ***FPIK** value.
- A ***Str** value, which passes up to 72 characters into a field as one character.

You can specify up to three default values, separated by commas, as the defaults for double and triple data types (for example, **Typ_Real2**, **Typ_Real3**).

Examples

Typ_Def_*GET(*Entity*,*ENTNUM*, *Item1*,*IT1NUM*,*Item2*,*IT2NUM*)	Gets a value the same way as the ANSYS ***GET** command
Typ_Def_*PAR(*Parm*)	Evaluates *Parm*, which is an ANSYS parameter
Typ_Def_*PICK(*N*)	Gets the field *N* information from the **Typ_XYZ_WP** picker, where *N* is a number from 1 to 9.
Typ_Def_*PAR(_z1),***PICK**(3),5	Used for a triple data type
Typ_Def_*CPAR(101)	Use the 101 value in the CSET vector

Typ_File

Creates the file selection box, which displays the filter, directory, and file name.

Function File Control

Explanation

Using this control produces a standard selection box on a dialog. The user can select a file, browse directories to find a file, or filter the contents of a directory according a user-specified extension.

Typ_File_Inline

Creates a text entry field for the directory, file name and extension.

Function File Control

Explanation

Creates a text entry field. When the user enters a string, it is converted automatically into the ANSYS format of *Filename,Extension,Directory*. The file name is limited to 250 characters including path, and the extension to 8 characters.

Typ_Idx

Creates an indexed list for side-by-side display of categories and their associated items.

Function Listing Control

Argument Description

Idx_*Category,Item,Value*

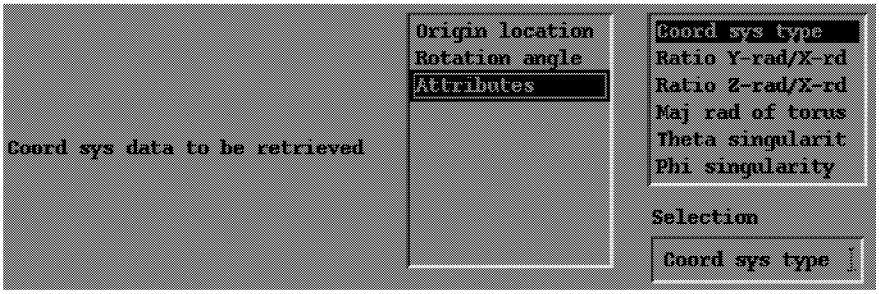
Defines each line of the scrolling lists. You must identify each item using the **Idx_** designator, followed by the category, item, and value. All categories are shown once in the left scrolling list. Each time a user chooses a category, all the items associated with that category appear in the right scrolling list. When a user chooses a specific item, the name of the item appears in a field labeled Selection.

Example

The following example creates a pair of scrolling lists permitting a user to select coordinate data about the model:

```
Fld_2
  Prm_Coord sys data to be retrieved
  Typ_IDX
  IDX_Origin location ,X               ,LOC,X
  IDX_Origin location ,Y               ,LOC,Y
  IDX_Origin location ,Z               ,LOC,Z
  IDX_Rotation angle  ,XY              ,ANG,XY
  IDX_Rotation angle  ,YZ              ,ANG,YZ
  IDX_Rotation angle  ,ZX              ,ANG,ZX
  IDX_Attributes      ,Coord sys type  ,ATTR,KCS
  IDX_Attributes      ,Ratio Y-rad/X-rd,ATTR,PAR1
  IDX_Attributes      ,Ratio Z-rad/X-rd,ATTR,PAR2
  IDX_Attributes      ,Maj rad of torus,ATTR,PAR1
  IDX_Attributes      ,Theta singularit,ATTR,KTHET
  IDX_Attributes      ,Phi singularity ,ATTR,KPHI
```

Here is the function block as it would appear in a dialog if a user had selected the **Coord sys type** item from the **Attributes** category:



Typ_Int

Creates a text entry field for displaying integer data.

`Function Numeric Control`

Notes

Using this command does not prevent non-integer data from being entered. **Typ_Int** takes the default and converts it to an integer before displaying it.

Typ_Int2

Creates two side-by-side text entry fields for displaying integer data.

`Function Numeric Control`

Notes

Using this command does not prevent non-integer data from being entered. **Typ_Int** takes the default and converts it to an integer before displaying it.

Typ_Int3

Creates three side-by-side text entry fields for displaying integer data.

```
Function Numeric Control
```

Notes

Using this command does not prevent non-integer data from being entered. **Typ_Int** takes the default and converts it to an integer before displaying it.

Typ_Lab

Defines a label data type which is used to display labels without an associated button or entry field.

```
Function Field Control
```

Explanation

Specify the labels using the **Prm_** control. **Typ_Lab** usually is used with the **Fld_**0 control. You can use multiple **Fld_**0 controls in a data block.

Example

```
Fld_0
   Typ_lab
      Prm_ Enter part name
```

Typ_Lis

Displays a list of choices for a field.

```
Function Field Control
```

Explanation

You can specify a number of types of lists:

- Single-selection Scrolling Lists

  Typ_Lis
  Lis_*Item,Value*

  Use one **Lis_** designator for each item on the list. Each item name can be 16 characters long, and will be shown in a scrolling window. The associated value of an item will be used by the command field.

- Multiple-selection Scrolling Lists

  Typ_MLis
  Lis_*Item,Value*

  Use one **Lis_** control for each item. The item name can be up to 16 characters long, and is shown in a scrolling window. The item's associated value is used by the command field. See the **Bnd_** command to learn how to limit the number of choices. Each item selected is separated by a comma.

- Radio Buttons

  Typ_Lis_RadioB
  Lis_*Item*,*Value*

  Use one **Lis_** designator for each item. The item name is shown as a radio button. The associated value of an item is used by the command field.

- Option Buttons

  Typ_Lis_OptionB
  Lis_*Item*,*Value*

  Use one **Lis_** designator for each item. The list of items is shown as a single option button. The associated value of an item is used by the command field.

- Database Read, Single-selection, or Multiple-selection Scrolling List

  Typ_Lis *or* **Typ_MLis**
  Lis_*READ,*Item*

  Current database items will be displayed in a scrolling window, from which the user can select one or more items. The selected item(s) will be used by the command field. *Item* will have one of the following values:

DOF	Current degree-of-freedom set
F	Current force set
CM	Current defined components
ASSM	Current defined assemblies
ETAB	Current element table items
PATH	Current path items
DV	Current design variables
SV	Current state variables
OBJ	Current objective function
PAR	Current parameters
PARX	Current parameters *not* used as opt. variables
FIT	Current fitting functions
MAT	Current material numbers
DPTH	Current list of paths
TYPE	Current element types
REAL	Current real constant numbers

Typ_Logi,Off_*Label*,On_*Label*

Creates an on/off toggle button.

```
Function Logical Control
```

Argument Description

Off_*Label*,
An optional character string containing up to 16 characters.

On_*Label*,
An optional character string containing up to 16 characters.

Typ_Real

Creates a text entry field for real number display.

```
Function Numeric Control
```

Argument Description

```
Fld_2
 Typ_Real
 Prm_Enter the width
```

Typ_Real2

Creates two side-by-side text entry fields for real number display

```
Function Numeric Control
```

Argument Description

```
Fld_2
 Typ_Real2
 Prm_Enter the X and Y offset
```

Typ_Real3

Creates three side-by-side text entry fields for real number display.

```
Function Numeric Control
```

Argument Description

```
Fld_2
 Typ_Real3
 Prm_X,Y,Z starting location
```

Typ_Resu

Enables the user to view results interactively on the screen by picking.

`Function Picking Control`

Explanation

To pass a parameter to the It*N* descriptor, use ***CPAR**(*N*). The IT vector is an A32 data type, and IT0_*String* will set the entire vector. The entity can be a node or an element, and the *Item* and *Component* are any items and components that can be plotted.

Specify the entity to be shown as IT1_*Entity*, the item to be shown as IT2_*Item*, and the component to be shown as IT3_*Component*.

Typ_Sep

Displays a separator bar in a dialog.

`Function Data Control`

Notes

This control should follow a **Fld**_0 control. Do not use with prompt mode.

Typ_XYZ

Obtains the coordinates of a point in picking mode.

`Function Picking Control`

Notes

Use this control only with the **Inp_P** control mode. You also can use **Typ_XYZ** in conjunction with the **Pwr**_1 and **Typ_Def**_***FPIK** UIDL function controls.

Typ_XYZ_SCREEN

Obtains the screen coordinates of a point in picking mode.

`Function Picking Control`

Notes

You can use this control only with the **Inp_P** control mode. The picking dialog displays automatically.

Typ_XYZ_WP

Invokes the model picker which obtains the working plane coordinates of a point in picking mode and other geometry information based on Rub_.

`Function Picking Control`

Notes

The data is stored in the PICK vector; you can extract it to a command in a chained function block by using **Typ_Def_*PICK**(*N*). You must use **Typ_XYZ_WP** with the **Typ_Def_*PICK** function control. You can pick up to six points.

---

*UILIST,*Fname,Ext,Dir*

Displays a named file in the ANSYS GUI lister window.

`GUI Command`

Argument Description

Fname
The name of the file.

Ext
The extension of that file.

Dir
The directory where the file resides. *Dir* defaults to the current working directory. Enclose the directory name in single quotation marks (for example, **'mydir'**).

Notes

You must use this command with the **Cmd_** function command control.

:X *INDEX*

UIDL processor inserts this line at the end of every control file.

`UIDL processor-inserted Command`

Notes

Do not use this line in blocks. ANSYS inserts this line *automatically* at the end of every control file. ANSYS also writes the contents of each **:N** command line, and shows the byte count of its location in the control file.

Appendix B. ANSYS Keywords

B.1. List of Keywords and Settings

Table B.1 Global Keywords

Command/Status	Keyword	Val	Keyword	Val	Comments
ANSYS Entry					
	BEGIN	1			
	"SYSTEM"	1			system name
	IEEE	1/0			system dependent
	X11	1			If-dX11
	X11C	1			If-dX11C
	X3D	1			If-dX3D
	LEVEL	1/0			GUI on or off
ANSYS Exit					
/EXIT,(blank)	RESULTS	0			
/EXIT,MODEL	RESULTS	0			
/EXIT,SOLU	N/A				
/EXIT,ALL	N/A				
/EXIT,NOSAV	N/A				
/SHOW					
/SHOW,X11	X11	1	X11C&X3D	0	
/SHOW,X11C	X11C	1	X11&X3D	0	
/SHOW,X3D	X3D	1	X11&X11C	0	
Routine Entry					
/PREP7	PREP7	1	BEGIN	0	Other routine names = 0
/SOLU	SOLUTION	1	BEGIN	0	Other routine names = 0
/POST1	POST1	1	BEGIN	0	Other routine names = 0
/POST26	POST26	1	BEGIN	0	Other routine names = 0
/OPT	OPT	1	BEGIN	0	Other routine names = 0
/RUNSTAT	RUNSTAT	1	BEGIN	0	Other routine names = 0
/AUX2	AUX2	1	BEGIN	0	Other routine names = 0
/AUX12	AUX12	1	BEGIN	0	Other routine names = 0
/AUX15	AUX15	1	BEGIN	0	Other routine names = 0
Routine Exit					
FINISH	BEGIN	1	"ROUTINE"	0	ROUTINE refers to above items
/EOF	BEGIN	1	"ROUTINE"	0	
/QUIT	BEGIN	1	"ROUTINE"	0	
SAVE					
SAVE		Write all active keywords to .db file			
RESUME					

Command/Status	Keyword	Val	Keyword	Val	Comments
RESUME		1. Read keyword settings from .db file 2. Re-establish the following keywords from the current stat of ANSYS: "routine" = 1 (currentroutinename) "system" = 1 (currentsystemname) LEVEL = 1/0 (GUI on or off)			
CLEAR					
/CLEAR		1. Set all keywords to zero 2. Re-establish the following keywords from the current stat of ANSYS: "routine" = 1 (currentroutinename) "system" = 1 (currentsystemname) LEVEL = 1/0 (GUI on or off)			

Table B.2 Discipline and Preference Keywords

Command/Status	Keyword	Val	Comments
ET,ETDELE,DOF			
	STRUCTRL	1	If DOFGRP = n0000 (n > 0)
	THERMAL	1	If DOFGRP = 0n000 (n > 0)
	ELECTRIC	1	If DOFGRP = 00n00 (n > 0)
	MAGNETIC	1	If DOFGRP = 000n0 (n > 0)
	FLUID	1	If DOFGRP = n000n (n > 0)
	CFD	1	See Element Specific Keywords
	MULTIFLD	1	Any combination of two or more of the above
Preferences Dialog Box			
	PR_STRUC	These are set to a value of one if they are chosen in the "Preferences" dialog box.	
	PR_THERM		
	PR_ELMAG		
	PR_FLUID		
	PR_CFD		
	PR_MULTI		
	PR_SET		
/PMETHOD	PMETHOD	1	If P-Method is turned on

Table B.3 Analysis Type and Option Keywords

Command/Status	SOPTCM Variable	Val	Keyword	Val	Comments
DEFAULT					
	ANTYPE	0	STATIC	1	
	ANCONT	0	NEWANLY	1	

Command/Status	SOPTCM Variable	Val	Keyword	Val	Comments
ANTYPE,					
STATIC	ANTYPE	0	STATIC	1	other "ANTYPE" & "options" = 0
BUCKL	ANTYPE	1	BUCKLING	1	
MODAL	ANTYPE	2	MODAL	1	
HARMIC	ANTYPE	3	HARMONIC	1	
TRANS	ANTYPE	4	TRNSIENT	1	
SUBST	ANTYPE	7	SUBSTRUC	1	
SPECT	ANTYPE	8	SPECTRUM	1	
NTYPE,,					
NEW	ANCONT	0	NEWANLY	1	RESTART = 0
RESTA	ANCONT	1	RESTART	1	NEWANLY = 0
BUCOPT,					
REDUC	EXTOPT	0	REDUCED	1	other "options" = 0
SUBSP	EXTOPT	1	SUBSPACE	1	
HROPT,					
REDUC	HRMOPT	1	REDUCED	1	other "options" = 0
FULL	HRMOPT	0	FULL	1	
MSUP	HRMOPT	2	MODESUP	1	
MODOPT,					
REDUC	EXTOPT	0	REDUCED	1	other "options" = 0
SUBSP	EXTOPT	1	SUBSPACE	1	
UNSYM	EXTOPT	3	UNSYMMET	1	
DAMP	EXTOPT	4	DAMPED	1	
TRNOPT,					
REDUCED	TRNOPT	1	REDUCED	1	other "options" = 0
FULL	TRNOPT	0	FULL	1	
MSUP	TRNOPT	2	MODESUP	1	
SPOPT,					
SPRS	SPANAL	0	SINGLEPT	1	other "options" = 0
DDAM	SPANAL	1	DDAM	1	
MPRS	SPANAL	2	MULTIPNT	1	
PSD	SPANAL	3	PSDRANDM	1	
EXPASS,					
ON	SRPASS	1			other "options" = 0
OFF	SRPASS	0			

Table B.4 Solution/Results Keywords

Command/Status	Keyword	Val
SolutionComplete		
SOLVE or PSOLVE	LSTEPGT1	1

Command/Status	Keyword	Val
	RESULTS	1
Results in Database		
SET, SUBSET, FLREAD	RESULTS	1
ExitfromSOLUTION		
FINISH, /EOF, /QUIT	LSTEPGT1	0

Table B.5 Miscellaneous Keywords

Command/Status	Keyword	Val
ESHAPE,		
0	NOMIXMSH	0
Not 0	NOMIXMSH	1
FMAGBC,		
command issued	FMAGBC	1
otherwise	FMAGBC	0
MOPT,		
AMESH,RV51	MOPTRV51	1
AMESH,RV52	MOPTRV51	0
MeshTool	MESHTOOL	1

Table B.6 Element-Specific Keywords

Element Type	Keyword	Val	Comments
SOLID5			
	ELEM3D	1	
	SCALARP	1	If KEYOPT(1) = 0, 1, 10
	ANISO	1	
PLANE13			
	ELEM2D	1	
	VECTORP	1	If KEYOPT(1) = 0, 4, 6
	ANISO	1	
PLANE53			
	ELEM2D	1	
	VECTORP	1	
SOLID62			
	ELEM3D	1	
	VECTORP	1	
SOLID64			
	ANISO	1	
SOLID96			
	ELEM3D	1	
	SCALARP	1	If KEYOPT(2) = 0

Element Type	Keyword	Val	Comments
	VECTORP	1	If KEYOPT(2) = 1,2
SOLID97			
	ELEM3D	1	
	VECTORP	1	
SOLID98			
	ELEM3D	1	
	SCALARP	1	If KEYOPT(1) = 0, 1, 10
	ANISO	1	
INFIN110			
	ELEM3D	1	
	SCALARP	1	If KEYOPT(1) = 0
INFIN111			
	ELEM2D	1	
	SCALARP	1	If KEYOPT(1) = 0
	VECTORP	1	If KEYOPT(1) = 1
INTER115			
	ELEM3D	1	
	SCALARP	1	
	VECTORP	1	
FLUID131			
	CFD	1	
FLUID132			
	CFD	1	
FLUID141			
	CFD	1	
FLUID142			
	CFD	1	

B.2. Logic Used for Keyword Evaluation

GUI Area	Filtering Used
Element Type	
No preferences	K_FL(pr_set+pmethod+cfd)
Structural	K_FL(PR_STRUC+pr_multi+pmethod+cfd)
Thermal	K_FL(PR_THERM+pr_multi+pmethod+cfd)
Electromagnetic	K_FL(PR_ELMAG+pr_multi+pmethod+cfd)
ANSYS Fluid	K_FL(PR_FLUID+pr_multi+pmethod+cfd)
Multiple	K_FL(PR_MULTI+pmethod+cfd)
CFD FLOTRAN	K_FL(PR_CFD*CFD)
P-Method	K_FL(PMETHOD+pr_cfd+cfd)
Loads/Solution	

GUI Area	Filtering Used
Structural	:K(PR_STRUC*pr_set) :K#(STRUCTRL)
Thermal	:K(PR_THERM*pr_set) :K#(THERMAL)
Electromagnetic	:K(PR_ELMAG*pr_set) :K#(ELECTRIC*MAGNETIC)
ANSYS Fluid	:K(PR_FLUID*pr_set) :K#(FLUID)
CFD FLOTRAN	:K(PR_CFD*pr_set) :K#(CFD)
Results Data	
Utility Menu	:K#(POST1+RESULTS)
Dialog Box	:K#(POST1+RESULTS)
Structural	K_FL(STRUCTRL+multifld)
Thermal	K_FL(THERMAL+multifld)
Electromagnetic	K_FL((ELECTRIC*MAGNETIC)+structrl+thermal+fluid)
ANSYSFluid	K_FL(FLUID+multifld)
CFD FLOTRAN	K_FL(CFD)
All Disciplines	K_FL(MULTIFLD+(STRUCTRL*THERMAL*FLUID))
Results Operations/Categories	
Structural	:K(PR_STRUC*pr_set) :K#(STRUCTRL+RESULTS)
Thermal	:K(PR_THERM*pr_set) :K#(THERMAL+RESULTS)
Electromagnetic	:K(PR_ELMAG*pr_set :K#((ELECTRIC*MAGNETIC)+RESULTS)
ANSYS Fluid	:K(PR_FLUID*pr_set) :K#(FLUID+RESULTS)
CFD FLOTRAN	:K(PR_CFD*pr_set) :K#(CFD+RESULTS)

Appendix C. ANSYS Product Codes

Table C.1 ANSYS Product Codes

Product	Product Codes
ANSYS Multiphysics (Mechanical + Emag + FLOTRAN)	FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC
ANSYS Mechanical	FULL_ANS LINPLUS THERMAL MULTDISC
ANSYS Structural	FULL_ANS LINPLUS
ANSYS Professional	LINPLUS THERMAL PROFS
ANSYS FLOTRAN	FLOTRAN
ANSYS Emag	E3 ELECMAG
ANSYS PrepPost	PP FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC
ANSYS University	FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC
ANSYS ResearchFS	FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC
ANSYS ED	ED FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC
ANSYS LS-DYNA s (Dyna solver & PrepPost stand-alone)	FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC LSDYNA
ANSYS DYNAPrepPost s (PrepPost stand-alone)	PP FULL_ANS LINPLUS THERMAL E3 ELECMAG FLOTRAN MULTDISC LSDYNA

Appendix D. Testing and Troubleshooting

D.1. Testing and Verification

D.1.1. Testing a Building Block

After a building block is created, you must test it. Follow these testing steps:

1. Create a **TEST.GRN** control file (see Chapter 1 for control file information).
2. Copy the new building block(s) to **TEST2.GRN**.
3. Add the name **TEST2.GRN** to the end of the **menulist55.ans** file.
4. Run ANSYS and check the functionality of the new building block(s).

If the building block doesn't seem to be working inside the GUI, that block may contain an error. Edit **TEST2.GRN** and verify that the **:S** and **:I** commands contain 0s in the appropriate places (see Chapter 1 for more information).

To aid in testing ")" lines, use the command **KEYW**,QALOGKEY,1 to force the writing of the ")" lines to the ANSYS log file. Setting the QALOGKEY keyword allows you, for debugging purposes, to see which commands were passed to ANSYS.

You can use the **/DEBUG** command to display some information. Issue the command **/DEBUG**,2 to see what is being written to the output file. To see the substitution of parameters, issue the command **/DEBUG**,4.

D.1.2. Debugging and Keywords

To see the value of keywords, type **keyword,stat** in the ANSYS Input window. To see the value of product code words, type **prdw,stat**. To see the values of GUI parameters, type ***stat,**_parameter*.

To eliminate keyword or product code filtering during debugging, set the **NOFILTER** flag.

Filtering only works in the first level of the Utility Menu--filtering does not work in cascaded menus.

Do not use a **!** and a blank on a line without additional characters.

Do not edit the control file once it's been indexed by ANSYS. Always edit the clean copy, as described in Chapter 1.

D.1.3. Verifying a Building Block

To verify a building block, examine the ANSYS log file. Make sure that the building block passed the correct information to ANSYS for *every* option available in that dialog box.

If your menus do not appear as expected, you might have not set up the control file header properly. Be sure both the header for the whole granule and the header for each individual block is set as documented.

To compare block design and functionality, copy existing building blocks into your test file and run tests on them.

D.2. Troubleshooting

D.2.1. Troubleshooting Dialogs

Dialog does not appear when selected.

The dialog may not be properly indexed. Check the function control file--If the **:S** command still has 0s, it means that block is ignored by ANSYS. Check to make sure the **:E END** command appears at the end of the block.

Index

Symbols

A

B

C

D

E

F

R

S

T

U

V

W

X