The Menu File

4

The menu file is a TXT file in the AutoCAD® *support* folder that defines most of the user interface. You can modify the menu file or create new menu files to add commands or macros to menus (including shortcut menus, image tile menus, and tablet menus) and to toolbars, assign commands to buttons on your pointing device, and create and modify shortcut keys.

In this chapter

Overview of Menu Files

The menu file is an ASCII TXT file consisting of sections that define the function of each part of the user interface except the command line, for example, pull-down menus, toolbars, and the buttons on a pointing device.

The default menu file is *acad.mnu*. You can find it in the *support* folder and open it in Notepad to see what a complete menu file looks like. To open the menu file, on the Tools menu, click Customize ➤ Edit Custom Files ➤ Current Menu.

You can create or modify menu files to

- Add or change menus (including shortcut menus, image tile menus, and tablet menus) and toolbars
- Assign commands to buttons on your pointing device
- Create and modify shortcut keys
- Add tooltips
- Provide Help text on the status line

To add a new menu, for example, you can modify the appropriate section of *acad.mnu* and save it under a new name, or you can create a new menu file.

In the following example, a drawing project requires frequent use of circles with a radius of 1, 2, or 3 units. To work more efficiently, you can create a menu file that defines a pull-down menu with three options, each of which draws a circle with a different radius. To create the menu file, you type the lines into Notepad (or any other text editor that saves in ASCII format) and then save the file in the AutoCAD *support* folder with an *.mnu* extension. The menu file in this example could be named *circles.mnu*.

```
***MENUGROUP=CIRCLES
***POP13
//Creates circles with radii 1, 2, 3
Circle-1 [Radius-1 ]^C^C_circle \1
Circle-2 [Radius-2 ]^C^C_circle \2
Circle-3 [Radius-3 ]^C^C_circle \3
```

The first line in a menu file is always the Menugroup section, which provides a unique name for the menu. In the example, the Menugroup name is Circles. The Menugroup name does not need to be the same as the file name.

The second line in the example is a section label. This menu is a pull-down menu and therefore uses a Pop section with a number from 1 through 16 (`POP13`). For more information about pull-down and shortcut menus, see "Create Pull-Down and Shortcut Menus" on page 70.

The third line, preceded by two slashes (//), is an optional comment line, which is ignored when the menu is compiled. You can use comments in menu files for copyright notices, documentation, or notes.

The next three lines define the items on the menu. In the first menu item, `Circle-1` is the name tag assigned to the menu item. The text in brackets, `[Radius-1 ]`, is the menu item label, which defines what is displayed on the menu for this item. The remainder is the menu macro, which uses AutoCAD commands and special characters to draw a circle with a radius of 1 unit. For information about creating menu macros, see "Create Menu Macros" on page 54.

In order to use the new menu in AutoCAD, you load the menu file, *circles.mnu*, with the MENULOAD command. For more information about loading menu files, see "Load and Unload Menu Files" on page 50.

Menu File Structure

Menu files are divided into sections. The Menugroup section is always first, and it assigns a unique menu group name to the menu file. A menu group name is a string of up to 32 alphanumeric characters with no spaces or punctuation marks.

The subsequent sections define specific areas of the AutoCAD interface and contain menu items, which usually consist of a name tag, a label, and a menu macro. For information specific to each section, see the topic for that section.

Menu file sections are identified by section labels that use the format `***section_name`. The multiple Buttons, Aux, Pop, and Tablet sections are numbered, for example, `***POP5`.

Section labels and associated elements in the user interface

Section label	User interface area
***MENUGROUP	Menu group name
***BUTTONS*n*	Pointing-device button menu
***AUX*n*	System pointing device menu
***POP*n*	Pull-down and shortcut menus
***TOOLBARS	Toolbar definitions
***IMAGE	Image tile menus
***SCREEN	Screen menus

Section labels and associated elements in the user interface (*continued*)

Section label	User interface area
***TABLET*n*	Tablet menus
***HELPSTRINGS	Text that is displayed in the status bar when a pull-down or shortcut menu item is highlighted, or when the cursor is over a toolbar button
***ACCELERATORS	Shortcut (or accelerator) key definitions

A menu file does not need to include every possible menu section. It is recommended that you create small menu files that can be loaded and unloaded on demand (with MENULOAD and MENUUNLOAD). Working with smaller files gives you better control of your system resources and makes customization easier.

Menu Items

The syntax that you use to create menu items is the same for all menu sections that use menu items. Each menu item can consist of a name tag, a label, and a menu macro. (Some sections do not use name tags, and some do not use labels.)

Name tag — Identifies the menu item. A menu item name tag is a string of alphanumeric and underscore (_) characters that uniquely identifies an item within a menu group.

Label — Defines what is displayed or presented to the user. The label is contained within square brackets ([and]).

Menu macro — Defines the action the menu item performs. Menu macros also define, for example, the appearance and location of toolbar buttons. Menu macros can be simple recordings of keystrokes that accomplish a task or a complex combination of commands and programming code.

A menu item normally resides on one line of the menu file and has the following format.

```
name_tag label menu_macro
```

In the following example from a Pop section, `ID_Quit` is the name tag. The label, `[Exit]`, displays Exit in the menu. When this menu item is selected, the menu macro, `^C^C_quit`, cancels any running commands and starts the QUIT command.

```
ID_Quit [Exit]^C^C_quit
```

Menu Item Labels

The format and use of menu item labels differ for each menu section. Menu sections that have no interface for displaying information (for example, the Buttons, Aux, and Tablet sections) do not require labels; however, labels can be used for internal notes in these sections. The following table describes how menu item labels are used in different sections of the menu file.

Use of menu section labels

Menu section	Use of label
POP*n*	Defines the content and format of pull-down and shortcut menus
TOOLBARS	Defines the toolbar name, status (floating or docked and hidden or visible), and position; also defines each button and its properties
IMAGE	Defines the text and image displayed in the image tile menus
SCREEN	Defines the text displayed in the screen menus
HELPSTRINGS	Defines the status line Help related to menu items in the Pop and Toolbars sections
ACCELERATORS	Associates keyboard action with menu macros

Menu Macros

A menu macro defines the action that results when a menu item is chosen. You can use commands, special characters, and DIESEL or AutoLISP programming code to create a menu macro. If you intend to include a command in a menu item, you must know the sequence of prompts and the default options for each.

Note As AutoCAD is revised and enhanced, the sequence of prompts for various commands (and sometimes even the command names) might change. Therefore, your custom menus might require minor changes when you upgrade to a new release of AutoCAD.

See Also

"Load and Unload Menu Files" on page 50
"Create Menu Macros" on page 54
"Create Pull-Down and Shortcut Menus" on page 70
"Customize Toolbars" on page 82
"Create Image Tile Menus" on page 87
"Create Tablet Menus" on page 96
"Create Status Line Help Messages" on page 97
"Create Shortcut Keys" on page 98
"Create Shortcut Keys" on page 98

Load and Unload Menu Files

Before you can use a menu file, it must be loaded into the program.

The *base menu* is automatically loaded when you start AutoCAD. In AutoCAD, the default base menu file is *acad.mnu*, and it resides in the AutoCAD *support* folder. If you modify the default menu or create a new menu file that you want to use as the base menu, you use MENU to load it. When you start AutoCAD again, the new base menu is automatically loaded.

The term *partial menu* refers to any menu file loaded after the base menu. You can use MENULOAD and MENUUNLOAD to load and unload partial menus as you need them during an AutoCAD session.

Any menu file can serve as a base menu or a partial menu, but it is recommended that you use a menu file that includes most sections as the base file and load additional smaller menu files as needed.

Load Menu Files

You can use MENULOAD and MENUUNLOAD to load and unload partial menus and add or remove individual pull-down menus from the menu bar.

AutoCAD stores the name of the last loaded base menu in the system registry. This name is also saved with the drawing, but it is used only for backwards compatibility. When you start AutoCAD, the last base menu used is loaded.

Change or Remove Menus

Frequent changes to the contents of a menu bar can be confusing. It is not recommended that you change the state of the menu bar visually except on explicit request. For example, if someone wants to unload an application, menus referenced specifically by that application could be removed as well.

To completely reinitialize the menu, remove all partial menus that are currently loaded by executing MENULOAD and selecting Replace All in the Menu Customization dialog box. This procedure removes all partial menus as well as their associated tag definitions and is equivalent to specifying a new menu file on the Files tab of the Options dialog box.

Restore or Alternate Menus

You can use a customized menu for some tasks while keeping the standard menu easily available. To load your custom menu, on the System tab of the Options dialog box, enter the custom menu name next to Menu File.

When you use MENULOAD or MENUUNLOAD to alter the loaded menus or customize the menu bar with Pop and Toolbar menus, the changes are saved to the registry. The next time you start AutoCAD, the menus that were loaded last and the menu bar configuration are restored. You can load and unload up to 8 partial menus and up to 16 Pop menus.

Work with Menu File Types

When you edit or create an MNU file, the next time it is loaded, AutoCAD compiles it and generates certain files in the AutoCAD *support* folder. The term *menu file* is used to refer to any of the group of files that work together to define the user interface, as described in the following table. When you load or unload a menu file, the Select Menu File dialog box may list MNU, MNC, or MNS files. You can select any of these files to load the menu you want.

The menu file types and their origins are described in the table.

Menu file types

File type	Description
MNU	Original ASCII menu file, the file you normally edit or create.
MNC	Compiled menu file; a binary file that contains the command strings and menu syntax that define the functionality and appearance of the menu or other interface element. AutoCAD compiles this file when you load an MNU file for the first time.
MNR	Menu resource file; a binary file that contains the bitmaps used by the menu or other interface element. AutoCAD generates this file each time it compiles an MNC file.
MNS	Source menu file; an ASCII file that is the same as the MNU file but does not include comments or special formatting. AutoCAD modifies this file each time the contents of the menu file change.

Menu file types (*continued*)

File type	Description
MNT	Menu resource file. This file is generated only when the MNR file is unavailable, for example, read-only.
MNL	Menu LISP file; contains AutoLISP expressions that are used by the menu file. AutoCAD loads this file into memory when a menu file with the same file name is being loaded.

AutoCAD finds and loads the specified file according to the following sequence. This same sequence is used when AutoCAD loads a new menu.

1 AutoCAD looks for a menu source (MNS) file of the given name, following the library search procedure.

- If an MNS file is found, AutoCAD looks for a compiled menu (MNC) file of the same name in the same directory. If AutoCAD finds a matching MNC file with the same or later date and time as the MNS file, it loads the MNC file. Otherwise, AutoCAD compiles the MNS file, generating a new MNC file in the same directory, and loads that MNC file.
- If an MNS file is not found, AutoCAD looks for a compiled menu (MNC) file of the given name, following the library search procedure. If AutoCAD finds the MNC file, it loads that file.
- If neither an MNS nor an MNC file is found, AutoCAD searches the library path for a menu template (MNU) file of the given name. If this file is found, AutoCAD compiles an MNC file, generates an MNS file, and then loads the MNC file.
- If no files of the given name are found, AutoCAD displays an error message and prompts you for another menu file name.

2 After finding, compiling, and loading the MNC file, AutoCAD looks for a menu LISP (MNL) file, using the library search procedure. If AutoCAD finds this file, it evaluates the AutoLISP expressions within that file.

The *acad.mnl* file contains AutoLISP code used by the standard menu file, *acad.mnu*. The *acad.mnl* file is loaded each time the *acad.mnu* file is loaded.

Each time AutoCAD compiles an MNC file it generates a menu resource (MNR) file, which contains the bitmaps used by the menu, and an MNS file, an ASCII file that is initially the same as the MNU file (without comments or special formatting). The MNS file is modified by AutoCAD each time you change the contents of the menu file through the interface (for example, when you use CUSTOMIZE to modify the contents of a toolbar).

Although the initial positioning of the toolbars is defined in the MNU or MNS file, changes to the show/hide and docked/floating status or changes to the toolbar positions are recorded in the system registry. After an MNS file has been created, it is used as the source for generating future MNC and MNR files. If you modify the MNU file after an MNS file has been generated, you must use the OPTIONS command to explicitly load the MNU file so that AutoCAD will generate new menu files and your changes will be recognized.

Note If you use the interface to modify the toolbars, you should cut and paste the modified portions of the MNS file to the MNU file before deleting the MNS file.

To load a partial menu

1. At the Command prompt, enter **menuload**.
2. In the Menu Customization dialog box, Menu Groups tab, enter the menu file name (or click Browse to select the file in the Select Menu File dialog box).
3. Click Load.
4. If the menu group includes pull-down menus, click the Menu Bar tab.
5. Under Menu Group, select the menu group you loaded.

   On the left, Menus lists the pull-down menus in the selected menu group. On the right, Menu Bar lists the menus currently displayed on the menu bar in order from left to right.
6. In the Menus list, select the menu you want to place on the menu bar.
7. In the Menu Bar list, select a menu and click Insert to place the new menu above the selected menu in the list..

   If no menu is selected in the Menu Bar list, the menu selected in the Menus list is inserted at the top of the Menu Bar list.
8. Click Close.

To unload a partial menu

1. At the Command prompt, enter **menuunload**.
2. In the Menu Customization dialog box, Menu Groups tab, select the menu file that contains the menus you want to unload.
3. Click Unload.
4. Click Close.

To load the base menu

1. At the Command prompt, enter **menu**.
2. In the Select Menu File dialog box, select the menu file you want to load.
3. Click Open.

Create Menu Macros

An item in the MNU file can have three parts: name tag, label, and menu macro. The macro defines the action to be executed when a user chooses the menu item.

Overview of Menu Macros

The macro in a menu item can be as simple as a command:

```
Line [Line]line
```

The name tag is `Line`, the label is `[Line]`, and the menu macro is `line`, which starts the LINE command.

In the next menu item example, the menu macro, `^C^C_circle \1`, draws a circle with a radius of 1 unit.

```
Circle-1 [Radius-1]^C^C_circle \1
```

- The special characters `^C^C` cancel any running commands.
- The special character underscore (_) automatically translates the command that follows into other languages.
- The entry `circle` starts the CIRCLE command.
- The special character backslash (\) creates a pause for the user to specify the center point.
- The entry `1` responds to the prompt for the radius.

For a list of special characters that you can use in menu macros, see "Use Special Control Characters in Menu Macros" on page 58.

In addition to commands and special characters, you can use DIESEL (Direct Interpretively Evaluated String Expression Language) and AutoLISP. See "DIESEL" on page 103.

Cancel Running Commands

To make sure that you have no AutoCAD commands currently in progress before executing a menu macro, use `^C^C` at the beginning of the menu macro. This is the same as pressing ESC twice. Although a single `^C` cancels

most commands, ^C^C is required to return to the Command prompt from a dimensioning command; therefore, it is good practice to use ^C^C.

Terminate Menu Macros

Every character in a menu macro is significant, even a blank space.

When a menu item is selected, AutoCAD places a space at the end of the macro before processing the command sequence. AutoCAD processes the macro in the following menu item as though you had entered **line** and then pressed SPACEBAR to complete the command:

```
Line [Line]line
```

In some situations, macros require special terminators. Some commands, for example, TEXT, must be terminated by pressing ENTER, not SPACEBAR. Some commands require more than one space (or ENTER) to complete, but some text editors cannot create a line with trailing blanks.

Two special conventions get around these problems.

- A semicolon (;) in a menu macro issues ENTER.
- If a line ends with a control character, a backslash (\), a plus sign (+), or a semicolon (;), AutoCAD does not add a blank space after it.

If the menu item in the following example simply ended with the backslash (which pauses the macro for user input), it would fail to complete the ERASE command, because AutoCAD does not add a blank after the backslash. Therefore, the menu macro uses a semicolon (;) to issue ENTER after the user input.

```
Erase 1 [Erase 1]erase \;
```

Compare the following menu items:

```
UCS [UCS ]ucs
UCS World [UCS W ]ucs ;
```

The first example enters **ucs** on the command line and presses SPACEBAR. The following prompt is displayed.

Origin/ZAxis/3point/Entity/View/X/Y/Z/Prev/Restore/Save/Del/?/ <World>:

The second example enters **ucs**, presses SPACEBAR, and presses ENTER, which accepts the default value, World.

Warning! As AutoCAD is revised and enhanced, the sequence of prompts for various commands (and sometimes even the command names) might change. Therefore, your custom menus might require minor changes when you upgrade to a new release.

Most menu macros work equally well in all sections of the menu file.

Suppress Echoes and Prompts in Menu Macros
Normally, characters that are read from a menu macro appear in the command window just as if you had entered them through the keyboard (that is, they are echoed). Prompts are displayed even if a menu item provides the responses. You can suppress these displays using the MENUECHO system variable. If echoes and prompts from menu item input are turned off, a `^P` in the menu item turns them on.

Create Long Menu Macros
If an item in the menu macro does not fit on one line, you can continue it on the next line. To do this, add a plus sign (+) as the last character of the line to be continued. The example below, which you might use to set initial conditions for a new drawing, continues onto a second line.

```
[Setup ]layer set ground-floor;;grid on; ... ;fill off;+
limits 0,0 12,9;status
```

Menu items can continue on as many lines as necessary.

Note Line breaks are not preserved when AutoCAD creates the MNS file.

See Also

"Use Special Control Characters in Menu Macros" on page 58
"Pause for User Input in Menu Macros" on page 56
"Provide International Language Support in Menu Macros" on page 58
"Repeat Commands in Menu Macros" on page 61
"Create Pull-Down and Shortcut Menus" on page 70
"Customize Toolbars" on page 82
"Create Image Tile Menus" on page 87
"Create Screen Menus" on page 91
"Create Tablet Menus" on page 96
"Create Status Line Help Messages" on page 97
"Create Shortcut Keys" on page 98
"DIESEL" on page 103

Pause for User Input in Menu Macros

To accept input from the keyboard or the pointing device in the middle of a menu macro, place a backslash (\) at the point where you want input.

```
Circle-1 [Circle-1]circle \1
Layoff [Layoff ]-layer off \;
```

The menu macro in Circle-1 pauses for the user to specify the center point and then reads a radius of 1. Note that there is no space after the backslash. The Layoff menu item starts LAYER on the command line, enters the Off option, and pauses for the user to enter one layer name. Layoff then turns that layer off and exits LAYER. LAYER normally prompts for another operation and exits only if you press SPACEBAR or ENTER. In the menu macro, the semicolon (;) is used for ENTER.

Normally, a menu macro resumes after one user input; for example, one point location. Therefore, you cannot construct a menu macro that accepts a variable number of inputs (as in object selection) and then continues. However, an exception is made for SELECT: a backslash suspends the menu item until object selection has been completed. Consider the following menu item example:

```
Make Red [Make Red ]select \change previous ;properties color red ;
```

In this menu item, SELECT creates a selection set of one or more objects. The macro then starts CHANGE, references the selection set using the Previous option, and changes the color of all selected objects to red.

Note Because the backslash character (\) causes a menu macro to pause for user input, you cannot use a backslash for any other purpose in a menu macro. When specifying file directory paths, use a forward slash (/) as the path delimiter: for example, /direct/file.

The following circumstances delay resumption of a menu macro after a pause:

- If input of a point location is expected, object snap modes may be used before the point is specified.
- If X/Y/Z point filters are used, the menu item remains suspended until the entire point has been accumulated.
- For SELECT only, the menu macro does not resume until object selection has been completed.
- If the user responds with a transparent command, the suspended menu macro remains suspended until the transparent command is completed and the originally requested input is received.
- If the user responds by choosing another menu item (to supply options or to execute a transparent command), the original macro is suspended, and the newly selected item is processed to completion before the suspended macro is resumed.

Note When command input comes from a menu item, the settings of the PICKADD and PICKAUTO system variables are assumed to be 1 and 0, respectively. This preserves compatibility with previous releases of AutoCAD and makes customization easier because you are not required to check the settings of these variables.

Provide International Language Support in Menu Macros

If you develop menu files that can be used with a non-English-language version of AutoCAD, the standard AutoCAD commands and options are translated automatically if you precede each command or option with the underscore character (_).

The following example shows a portion of a Pop menu.

```
[->Arc]
[3-point]^C^C_arc
[Start, Cen, End]^C^C_arc;\_c
[Start, Cen, Angle]^C^C_arc;\_c;\_a
[Start, Cen, Length]^C^C_arc;\_c;\_l
[Start, End, Angle]^C^C_arc;\_e;\_a
[Start, End, Radius]^C^C_arc;\_e;\_r
```

In the example, the underscore precedes every command or an option used in the menu macros.

Use Special Control Characters in Menu Macros

Special characters, including control characters, can be used in menu macros. In a menu macro, the caret (^) maps to the CTRL key on the keyboard. You can combine the caret with another character to construct menu macros that do such things as turn the grid on and off (^G) or cancel a command (^C). Because brackets ([and]) identify menu labels, they cannot be used in menu macros.

```
[GridFlip]^G
[*Cancel*]^C
```

The nonalphabetic control characters are as follows:

```
^@ (ASCII code 0)
^[ (ASCII code 27)
^\ (ASCII code 28)
^] (ASCII code 29)
^^ (ASCII code 30)
^_ (ASCII code 31)
```

The macro in the Address menu item below uses the backslash (\) to pause for user input and the semicolon (;) for ENTER.

```
Address [Address ]text \.4 0 DRAFT Inc;;;Main St.;;;City, State;
```

The macro starts TEXT, pauses for the user to specify a start point, and then enters the address on three lines. In the triple semicolon (;;;), the first semicolon ends the text string, the second repeats TEXT, and the third accepts the default placement below the previous line.

You may want a menu macro to enter one or more characters but not submit them as final input. For example, you could create a series of menu macros to act as a numeric keypad.

```
[1]1x^H
[2]2x^H
[3]3x^H
```

When you choose one of these items, the appropriate digit is entered. Another character follows (the letter *x* in this case), and that character is removed by ^H. (CTRL+H is the ASCII code for a BACKSPACE.) Each of these menu items ends with a control character, and AutoCAD does not add a space or ENTER to such items. Thus, you can choose [2], [2], [3], [1] to construct the input 2231. Press ENTER to enter the completed number.

Menu macros use the special characters listed in the following table. Brackets ([and]) identify menu labels and cannot be used in menu macros.

Special characters used in menu macros

Character	Description
;	Issues ENTER
^M	Issues ENTER
^I	Issues TAB
[blank space]	Enters a space; a blank space between command sequences in a menu item is equivalent to pressing the SPACEBAR
\	Pauses for user input (cannot be used in the Accelerators section)
_	Translates AutoCAD commands and options that follow
+	Continues the menu macro to the next line (if last character)
=*	Displays the current top-level pull-down, shortcut, or image menu
*^C^C	Prefix for a repeating item
$	Loads a menu section or introduces a conditional DIESEL macro expression ($M=)
^B	Turns Snap on or off (CTRL + B)
^C	Cancels a command (ESC)
^D	Turns Coords on or off (CTRL + D)
^E	Sets the next isometric plane (CTRL + E)
^G	Turns Grid on or off (CTRL + G)
^H	Issues BACKSPACE
^O	Turns Ortho on or off
^P	Turns MENUECHO on or off
^Q	Echoes all prompts, status listings, and input to the printer (CTRL + Q)
^T	Turns tablet on or off (CTRL + T)
^V	Changes the current viewport
^Z	Null character that suppresses the automatic addition of SPACEBAR at the end of a menu item

See Also

"ASCII Codes" on page 149

Repeat Commands in Menu Macros

Once you have selected a command, you might want to use it several times before moving on to another command. In a menu macro, you can repeat a command until you choose another command. You cannot use this feature to choose options.

If a menu macro begins with `*^c^c` immediately following the item label, the macro is saved in memory. Subsequent Command prompts are answered by that macro until it is terminated by ESC or by selection of another menu item.

Note Do not use ^C (Cancel) within a menu macro that begins with the string *^C^C; this cancels the repetition.

The menu macros in the following examples repeat the commands:

```
[Move ]*^C^Cmove Single
[Copy ]*^C^Ccopy Single
[Erase ]*^C^Cerase Single
[Stretch]*^C^Cstretch Single Crossing
[Rotate ]*^C^Crotate Single
[Scale ]*^C^Cscale Single
```

Each of the macros in the example starts the command and prompts the user to select one object. Any other prompts necessary to complete the command are displayed, and then the command ends and starts again. For information about `Single` and `Single Crossing`, see "Use Single Object Selection Mode in Menu Macros" on page 61.

Command repetition cannot be used in menu macros for image tile menus.

Use Single Object Selection Mode in Menu Macros

Single Object Selection mode cancels the normal repetition of the Select Objects prompt in editing commands. After you select one object and respond to any other prompts, the command ends. Consider the menu macro in the following example:

```
[Erase]*^C^Cerase single
```

This macro terminates the current command and starts ERASE in Single Object Selection mode. After you choose this menu item, you either select a single object to be stretched or click a blank area in the drawing and specify window selection. Any objects selected in this way are erased, and the menu item is repeated (due to the leading asterisk) so that you can erase additional objects.

Use Menu Macros to Swap Menus

You can use menu macros to replace the contents of an active Buttons, Aux, Pop, Screen, or Tablet menu section. The new menu content can be from another section or submenu in the base menu, or it can come from a partial menu. Menu-swapping information that is related to each menu section is included with the description of that menu section.

You can swap menus only of the same type—one Aux for another, one Pop for another, and so on. Trying to swap between types may result in unpredictable and undesired behavior. However, within a given type, you can swap any menu for any other menu. Swapping can lead to some strange behavior for Tablet menus, because they typically do not all have the same number of macros.

Use the following syntax in a menu macro to swap menus:

```
$section=menugroup.menuname
```

These are the descriptions:

$	Loads a menu section.
section	Specifies the menu section. Valid names are `A1–A4` for Aux menus 1 through 4 `B1–B4` for Buttons menus 1 through 4 `P0–P16` for Pop menus 0 through 16 `I` for the Image menu `S` for the Screen menu `T1–T4` for Tablet menus 1 through 4
menugroup	Specifies the menu group that *menuname* is a member of (not necessary if *menuname* is in the base menu).
menuname	Specifies which section or submenu to insert. It is the main label or alias for the section to load.

The following menu items illustrate submenu referencing:

```
$S=PARTS
$T1=EDITCMDS
```

You can activate the submenu mechanism in the middle of a command without interrupting it. For example, the following command strings are equivalent:

```
$S=ARCSTUFF ARC
ARC $S=ARCSTUFF
```

Each menu item starts the ARC command, switches to the ARCSTUFF screen submenu, and awaits the entry of arc parameters. A space must follow the submenu reference to separate it from subsequent commands in the menu item.

A Pop menu can be present either in the menu bar or on the active shortcut menu but not both.

Use Conditional Expressions in Menu Macros

You can use the $M= command within a menu macro to introduce macro expressions written in DIESEL (Direct Interpretively Evaluated String Expression Language). The format is

```
$M=expression
```

Introducing the macro with $M= tells AutoCAD to evaluate the following string as a DIESEL expression, and that *expression* is the DIESEL expression.

```
[Fillflip]FILLMODE $M=$(-,1,$(getvar,fillmode))
```

Fillflip switches FILLMODE on and off by subtracting the current value of FILLMODE from 1 and returning the resulting value to the FILLMODE system variable. You can use this method to toggle system variables whose valid values are 1 or 0.

Termination of Macros That Contain Conditional Expressions

If you use the DIESEL string language to perform "if-then" tests, conditions might exist where you do not want the normal terminating space or semicolon (resulting in ENTER). If you add ^z to the end of the menu macro, AutoCAD does not automatically add a space (ENTER) to the end of the macro expression. See "Overview of Menu Macros" on page 54 and "DIESEL" on page 103.

As with other control characters in menu items, the ^z used here is a string composed of ^ (a caret) and z and is not equivalent to pressing CTRL+Z.

In the following examples, ^z is used as a menu macro terminator.

```
[Model]^C^C$M=$(if,$(=,$(getvar,tilemode),0),$S=mview _mspace )^Z
[Paper]^C^C$M=$(if,$(=,$(getvar,tilemode),0),$S=mview _pspace )^Z
```

If these menu macros did not end with ^z, AutoCAD would automatically add a space (ENTER), repeating the last command entered.

See Also

"Overview of Menu Macros" on page 54
"Use Special Control Characters in Menu Macros" on page 58
"DIESEL" on page 103

Use AutoLISP in Menu Macros

You can use AutoLISP variables and expressions to create menu macros that perform complex tasks. To use AutoLISP efficiently in menu macros, you can place AutoLISP code in a separate MNL file. AutoCAD loads the MNL file when it loads a menu file with the same name.

AutoCAD accepts up to 255 characters of AutoLISP code in menu macros. To use more characters, break up the code into separate modules separated by semicolons (;) so that AutoCAD can read and execute the code in blocks.

Creating menu items that use AutoLISP is a more advanced way to use the AutoCAD custom menu feature. Carefully study the following examples and the information in the *AutoLISP Reference* and the *AutoLISP Developer's Guide* (on the Help menu, click Developer Help). Experimentation and practice will help you use this feature effectively.

Call a Menu Macro

To programmatically execute a Pop menu macro as though the user chose it, you can use the following syntax:

```
(menucmd "Gmenugroup.name_tag=|")
```

This works only if the Pop menu macro is part of a Pop menu that is on the AutoCAD menu bar and available for use. For more information about this syntax, see the *AutoLISP Reference*.

Preset Values

An application that uses block insertion presets could provide menu items like these:

```
[Set WINWID ]^C^C^P(setq WWID (getreal"Enter window width: ")) ^P
[Set WALLTHK]^C^C^P(setq WTHK (getreal"Enter wall thickness: ")) ^P
[Insert Window]^C^C_INSERT window XScale !WWID YScale !WTHK
```

This code inserts the block named window, scaling its *X* axis to the current window width and its *Y* axis to the current wall thickness. In this example, the actual values come from the user-defined AutoLISP symbols WINWID and WALLTHK. The rotation is up to the user to decide so that the window can be rotated in the wall.

Resize Grips

With the following menu items, grip size adjustment can be done on the fly:

```
[GRIP-up]^P(setvar"gripsize"(1+(getvar"gripsize")))(redraw)(princ)
[GRIP-dn]^P(setvar"gripsize"(1-(getvar"gripsize")))(redraw)(princ)
```

To add validity checking to these menu items, values less than 0 and greater than 255 cannot be used for the GRIPSIZE system variable.

Prompt for User Input

The following menu item prompts the user for two points and draws a rectangular polyline with the specified points as its corners.

```
[BOX  ]^P(setq a (getpoint "Enter first corner: "));\+
(setq b (getpoint "Enter opposite corner: "));\+
pline !a (list (car a)(cadr b)) !b (list (car b)(cadr a)) c;^P
```

Customize Buttons on a Pointing Device

The Windows system pointing device uses the Aux menus in the MNU file; any other pointing device uses the Buttons menus.

Overview of Buttons and Aux Menus

The Buttons (***BUTTONS*n*) and Aux (***AUX*n*) sections of the menu file are identical in format. The Windows system pointing device uses the Aux menus, and any other pointing device (for example, a puck or stylus for a digitizing tablet) uses the Buttons menus. The BUTTONS1 menu functions identically to the AUX1 menu, and so on.

Buttons and Aux menu sections are only valid when the menu file is used as a base menu; Buttons and Aux sections are ignored in partial menus.

In the numbered Aux and Buttons sections of the menu file, each line represents a button. Your pointing device can recognize as many lines as it has assignable buttons. For example, in the AUX1 and BUTTONS1 sections, each line defines what happens when you click a particular button, and in the AUX2 and BUTTONS2 sections, each line defines what happens when you press SHIFT and click a particular button.

Buttons and associated menu sections

Key/button sequence	Menu sections
Click	AUX1 and BUTTONS1
SHIFT + click	AUX2 and BUTTONS2
CTRL + click	AUX3 and BUTTONS3
CTRL + SHIFT + click	AUX4 and BUTTONS4

Although sections 1 through 4 are the only active sections, you can define additional sections and swap them into the active sections. See "Swap Buttons and Aux Menus" on page 67.

The following AUX1 section example is similar to that found in the standard *acad.mnu* file:

```
***AUX1
;
^C^C
^B
^O
^G
^D
^E
^T
```

The first line after the menu section label, ***AUX1, represents the next button after the pick button on your pointing device (button 2). The semicolon (;) assigns ENTER to button 2. The second line after the menu section label assigns ESC twice to button 3, and so on.

Note The first line after the menu section label ***AUX1 or ***BUTTONS1 is used only when the SHORTCUTMENU system variable is set to 0. If SHORTCUTMENU is set to a value other than 0, the built-in menu is used. Similarly, the second line after the ***AUX1 or ***BUTTONS1 label is used only when the MBUTTONPAN system variable is set to 0.

You cannot reassign the pick button in the menu file. The pick button assignment is controlled by the operating system, or a device-specific configuration. The default pick button can be different on each pointing device, depending on the manufacturer.

Because labels in button menus are not displayed, you can use the labels as comments. The following example uses the label area to note the button number.

```
***AUX1
[button no.2];
[button no.3]$P0=*
[button no.4]^C^C
[button no.5]^B
```

The macro assigned to button number 3 in the example causes another menu to be displayed. It has the following format:

```
$Pn=*
```

The `$` is the special character code for loading a menu area; `Pn` specifies the Pop menu section; and `=*` displays what is currently loaded to the specified menu area.

Therefore, in the example from *acad.mnu*, clicking button number 3 displays the menu assigned to the P0 location. (The P0 menu location is the shortcut menu usually called from the Buttons or Aux menu. The P1 through P16 locations are left to right on the menu bar.) Typically, the Pop0 section of the menu file is assigned to the P0 location.

Each remaining line in that section assigns a command sequence to each subsequent button on the pointing device. For example, `^C^C` (ESC twice) is assigned to button 4, and `^B` (Snap mode toggle) is assigned to button 5.

See Also

"Create Menu Macros" on page 54
"Use Special Control Characters in Menu Macros" on page 58

Swap Buttons and Aux Menus

If necessary, you can swap the contents of the active Buttons and Aux menus with that of another menu section of the same type. In addition to the standard `AUX1` through `AUX4` (and `BUTTONS1` through `BUTTONS4`) sections, you can define additional numbered Aux and Buttons sections for specific purposes.

Note Even though section labels such as ***AUXTEST and ***BUTTONS1–2 are currently valid, numbered labels such as ***AUX10 or ***BUTTONS15 are preferred for their long-term compatibility.

These additional sections can use aliases with the syntax ***label*. The labels must come between the ****section* line and the first menu item line for that section. The alias *label* string can be any string. It does not need to contain any keyword. You can have as many aliases as you like for each section. The alias labels, as well as the section labels, can be used to identify the menu for swapping purposes. In the example below, Aux12 has two aliases; the section label or either of the aliases is sufficient to identify the menu.

For example, if you want to change the standard right-click action so that AutoCAD displays a different shortcut menu, you can use the following menu syntax.

```
***AUX2
// Shift + button
$P0=SNAP $p0=*

***AUX4
// Control + Shift + button - Toggles to custom A2
$A2=CUSTOM_A2 $A4=CUSTOM_A4 ^P(princ ">> Custom A2 <<")(princ) ^P

***AUX12
**CUSTOM_A2
**MYPOP
// Shift + button - Displays the MYPOP menu
$P0=MYPOP $p0=*

***AUX14
**CUSTOM_A4
// Control + Shift + button - Toggles back to default A2
$A2=AUX2 $A4=AUX4 ^P(princ ">> Default A2 <<")(princ) ^P
```

Replacing the standard Aux4 definition with the one shown in the example allows you to swap the content of the A2 and A4 menus. After reloading the MNU file, CTRL+SHIFT+click loads the contents of the menus defined by the alias CUSTOM_A2 into the A2 menu position and CUSTOM_A4 into the A4 position. Then, using AutoLISP, AutoCAD displays a message on the command line. Now when you SHIFT+click, the MYPOP menu is loaded into the P0 menu position and is used as the shortcut menu. To return to the default P0 menu, CTRL+SHIFT+click (which now calls the CUSTOM_A4 menu) to load the default AUX2 menu back into the A2 menu position and the AUX4 menu back into the A4 position.

You can also use the AutoLISP **menucmd** function to swap Aux and Buttons menus. Assuming that the previous examples are in an MNU file with a menugroup of MYGROUP, the following function call loads the CUSTOM_A2 menu into the A2 menu position.

```
(menucmd "A2=mygroup.custom_a2")
```

Note Previous AutoCAD releases allowed the ***label* syntax after menu items within a menu section; these were known as submenus. The submenu syntax is still accepted by AutoCAD, but it is converted into a ****section* label in the MNS file. This syntax is not guaranteed to be valid in future releases. It is recommended that you change all ***label* submenus into ****section* labels. The ***alias* label is valid, provided that it occurs after a section label and before any menu items.

Accept Coordinate Entry in Button Menus

When you click one of the buttons on a multibutton pointing device, AutoCAD reads not only the button number but also the coordinate of the crosshairs at the time you click. By carefully constructing the macros in the Buttons and Aux sections of the menu file, you can choose to either ignore the coordinate or use it with the command activated by the button.

As described in "Pause for User Input in Menu Macros" on page 56, you can include a backslash (\) in a menu item to pause for user input. For the Aux and Buttons menus, the coordinate of the crosshairs is supplied as user input when the button is clicked. This occurs only for the first backslash in the menu item; if the item contains no backslashes, the crosshairs coordinate is not used. Consider the following menu items:

```
***AUX2
line
line \
```

The first button starts the LINE command and displays the Specify First Point prompt in the normal fashion. The second button also starts the LINE command, but AutoCAD uses the current crosshairs location at the Specify First Point prompt and displays the Specify Next Point prompt.

Create Pull-Down and Shortcut Menus

Pull-down menus are pulled down from the menu bar. Shortcut menus (also called context menus) are displayed at or near the crosshairs or cursor in the drawing window, in the text window or command window, or in toolbar areas. Both are defined in the Pop sections of the MNU file.

Overview of Pull-Down and Shortcut Menus

The pull-down and shortcut menus are displayed as cascading menus (also known as walking or hierarchical menus).

There are two types of shortcut menus; both are displayed at the cursor location. The context menus, displayed when you right-click, provide commands appropriate to your current activity or the location of the cursor on the screen. The object snap menu, displayed when you hold down SHIFT and right-click, provides object snaps and tracking options.

Pull-down menus are defined in the `***POP1` through `***POP499` menu sections, and shortcut menus are defined in the `***POP0` and `***POP500` through `***POP999` sections. A pull-down menu can contain up to 999 menu items. A shortcut menu can contain up to 499 menu items. Both limits include all menus in a hierarchy. If menu items in the menu file exceed these limits, AutoCAD ignores the extra items. If a pull-down or shortcut menu is longer than the available display space, it is truncated to fit.

Pull-down menus are always *pulled down* from the menu bar, but the shortcut menu is always displayed at or near the crosshairs or cursor in the graphics area, in the text window or command window, or in the toolbar areas. The syntax for both of these `POP`*n* menu sections is the same except that the shortcut menu title is not included in the menu bar. The shortcut menu title is not displayed at all (but you must still enter a dummy title).

Access to the shortcut menu is through the `$P0=*` menu command, which can be issued by another menu item (such as a `BUTTONS`*n* menu item) or by an AutoLISP or ObjectARX® program. While the shortcut menu is active, the menu bar is not available.

Create Separator Lines on Menus

To create a horizontal line between items displayed on a menu, use a menu item label containing two hyphens:

```
[--]
```

The width of each pull-down and shortcut menu is determined by its widest label, and the separator line expands to fill the entire width of the menu. (A menu macro on a line with a separator line label is ignored.)

Create Cascading Submenus

Pull-down and shortcut menu item labels use special characters (such as –>, <–, and <–<–) to control the hierarchy of cascading menus. These special characters indicate submenus and last items in submenus and can also terminate all parent menus. Each special character string must come first in a menu item label.

The special character –> indicates that this item has a submenu, as in the following example:

```
[–>&Zoom]
```

If you pull down the View menu and click Zoom or move the cursor to the right end of the item, the Zoom submenu is displayed.

The special character <– indicates that the item is the last item in a submenu, as in the following example:

```
[<–&Extents]
```

Special characters <–<– indicate that the item is the last item of a submenu and also of its parent menu, as in the following examples of some labels from the Modify menu:

```
[&Modify]
[->&Object]
[->&Text]
[<-<-&Justify]
```

The first example is the label for the Modify menu. The label for Object uses a special character to indicate that it has a submenu. Text is part of the Object submenu and has a submenu of its own. Justify is the last item on the Text submenu and also ends the Modify menu.

The characters described in the following table are the only nonalphanumeric characters that can be used in a Pop section label. Nonalphanumeric characters not listed are reserved for future use as special menu characters.

Special characters for labels in Pop menu sections

Character	Description
`--`	Expands to become a separator line in the pull-down and shortcut menus (when used with no other characters).
`–>`	Indicates that the pull-down or shortcut menu item has a submenu.
`<–`	Indicates that the pull-down or shortcut menu item is the last item in the menu or submenu.
`<–<–`	Indicates that the pull-down or shortcut menu item is the last item in the submenu and terminates the parent menu. (One `<–` is required to terminate each parent menu.)
`$(`	Enables the pull-down or shortcut menu item label to evaluate a DIESEL string macro if `$(` are the first characters.
`~`	Makes a menu item unavailable.
`!.`	Marks a menu item with a check mark.
`&`	Placed directly before a character, specifies that character as the menu access key in a pull-down or shortcut menu label. For example, `S&ample` displays S<u>a</u>mple (with the letter *a* underlined).
`/c`	Specifies the menu access key in a pull-down or shortcut menu label. For example, `/aSample` displays S<u>a</u>mple (with the letter *a* underlined).
`\t`	Pushes all label text to the right of these characters to the right side of the menu.

Create Pull-Down Menus

The `POP1` through `POP499` menu file sections define menus that are pulled down from the menu bar.

The following example illustrates the syntax that is used to create a pull-down menu.

```
***POP13
**MYTOOLS
M_Tools [&MyTools]
M_Save  [&Save\tCtrl+S]^C^C_qsave
        [--]
M_ShwTB [Show MyToolbar]^C^C_-toolbar mytools s
M_HidTB [Hide MyToolbar]^C^C_-toolbar mytools h
        [--]
M_EMenu [Edit MyMenu]^C^C^P(command"notepad"(findfile"my.mnu")) ^P
M_LMenu [Reload MyMenu]^C^C^P(command"_menu""my.mnu") ^P
        [--]
M_EPgp  [Edit PGP]^C^C^P(command"notepad"(findfile"acad.pgp")) ^P
M_LPgp  [Reload PGP]^C^C_re-init 16
```

Each menu section can have one or more aliases that are defined by ***alias* labels following the ***POPn* menu section label. In the previous example, **MYTOOLS is an alias for the POP13 menu. For additional information about menu aliases, see "Swap Button and Auxiliary Menus" on page 30.

Note The *POPn* menu sections no longer support the ***submenu* syntax used in previous releases. The ***alias* syntax is valid, provided that it occurs after a section label and before any menu items.

For the POP1 through POP16 menu sections, AutoCAD constructs a menu bar containing the titles of those sections. If no POP1 through POP16 sections are defined, AutoCAD inserts default File and Edit menus.

Note If no active pull-down menus are defined (POP1 through POP16), the shortcut menu, POP0, does not function.

Pop menu sections numbered higher than POP16 and lower than POP500 are available to be inserted into the menu bar with MENULOAD or through the menu swapping process.

Define the Menu Bar Title

For pull-down menus, the first label defines the title that is displayed in the menu bar. The following example is the top portion of the POP2 pull-down menu section.

```
***POP2
ID_MnEdit [&Edit]
ID_U [&Undo\tCtrl+Z]_u
ID_Redo [&Redo\tCtrl+Y]^C^C_redo
```

On the first line after the section label, `***POP2`, the label `[&Edit]` causes Edit to be displayed as a menu bar title. The ampersand (&) preceding the letter *E* underlines that letter to indicate that it is the access key. The name tag associated with the menu title, `ID_MnEdit`, can be used to enable and disable this entire menu. (The line that defines the menu bar title cannot have a menu macro.)

To access a menu or a menu item from the keyboard, you hold down ALT and press the access key. Any letter in the label can be the access key, but an access key must be unique for its menu or submenu. For example, on the Modify menu in AutoCAD, *m* is used for Match Properties, so Mirror and Move must use other letters. The letter *t* can be used for both Trim and Text because Text is on the Object submenu.

See Also

"Swap Buttons and Aux Menus" on page 67
"Swap and Insert Pull-Down Menus" on page 80

Create Shortcut Menus

Shortcut menus are defined in the MNU file using the same syntax as pull-down menus. The `POP0` menu section defines the default Object Snap shortcut menu, and the menu sections from `POP500` through `POP999` are used for context shortcut menus.

AutoCAD references the context shortcut menus by their alias (for example, `**GRIPS`) and uses them in specific situations. The actual `POP`*n* number is not important, but the alias names must follow the proper naming conventions in order to be used. The following aliases are reserved for use by AutoCAD:

GRIPS	The content of this menu defines the Hot Grip shortcut menu (right-click in the drawing area while a grip on an object is selected).
CMDEFAULT	The content of this menu defines the Default mode shortcut menu (right-click in the drawing area while no command is active and no objects are selected).
CMEDIT	The content of this menu defines the Edit mode shortcut menu (right-click in the drawing area while one or more objects are selected, no grips are selected, and no command is active).

CMCOMMAND — The content of this menu defines the Command mode menu (right-click in the drawing area while a command is active). In addition to the content of the CMCOMMAND menu, the command line options (keywords within the square brackets) are inserted into this menu.

Context-Sensitive Shortcut Menus

The CMEDIT and CMCOMMAND shortcut menus can be made context-sensitive. In addition to the content of the CMEDIT menu, the appropriate object menu (if it exists) is inserted into this menu when one or more of a specific object type are selected. Object menus use the following naming convention: OBJECT_*objectname* or OBJECTS_*objectname*.

If a single object is selected, the OBJECT_*objectname* menu is used, and if more than one of the same object is selected, the OBJECTS_*objectname* menu is used. If no OBJECT_*objectname* is available, AutoCAD uses the OBJECTS_*objectname* menu (if it exists).

The object name is the DXF name of the object in all cases except the insert object. To differentiate between a block insertion and an xref, use the names BLOCKREF and XREF.

The following AutoLISP® code defines the command OTYPE, which reports the selected object's DXF name.

```
(defun C:OTYPE()
  (cdr (assoc 0 (entget (car (entsel))))))
```

For example, to support an object-specific shortcut menu item for one or more selected block references, insert the following:

```
***POP512
**OBJECTS_BLOCKREF
[shortcut menu for block objects]
ID_BLOCK [Explode] ^C^C_explode
```

Like the CMEDIT menu, the CMCOMMAND menu can have context-sensitive information added to it. Any menu named COMMAND_*commandname* is appended to the CMCOMMAND menu. The text of *commandname* can be any valid AutoCAD command, including any custom-defined or third-party commands.

To make this work with a hyphen-prefixed command (such as -INSERT), you need to name the menu COMMAND_-INSERT.

Control Display of Menu Item Labels

You can control the way that item labels are displayed. Display them as gray, making them unavailable to the user, or mark them with a check mark.

Labels can also contain DIESEL string expressions to modify the contents of the label. This can disable, mark, or interactively change the text of the displayed label. See "DIESEL Expressions in Menu Macros" on page 108.

When disabling and marking menu item labels, be sure to use an appropriate technique that keeps track of changes that affect the state of the label.

Disable Menu Item Labels

To gray out a menu item, begin the menu item label with a tilde (~). By convention, this indicates that the item is disabled, that is, not available. Any commands associated with the item are not issued, and any submenus are inaccessible.

For example, the following menu items are disabled.

```
[~Line]
[~->Pline]
```

Menu item labels can contain DIESEL string expressions that conditionally disable or enable menu item labels each time they are displayed. For example, the DIESEL string expression within the following menu item label disables the item while a command is active.

```
[$(if,$(getvar,cmdactive),~)MOVE]^C^C_move
```

The AutoLISP **menucmd** function can be used to disable and enable items from a menu macro or application. For examples, see "Reference a Pull-Down or Shortcut Menu" on page 77.

Mark Menu Item Labels

You can mark a menu item label with a leading check mark by including an exclamation point and a period (**!.**). Marking a menu item does not restrict your ability to choose the item, although a marked item can be disabled.

In the following example, the Line menu item is marked with a check.

```
[!.Line]
```

Menu item labels can contain DIESEL string expressions that conditionally mark menu item labels each time they are displayed. The following example places a check mark to the left of menu labels whose related system variables are currently enabled.

```
[$(if,$(getvar,orthomode),!.)Ortho]^O
[$(if,$(getvar,snapmode),!.)Snap]^B
[$(if,$(getvar,gridmode),!.)Grid]^G
```

The AutoLISP **menucmd** function can be used to mark labels from a menu macro or application. For examples, see "Reference a Pull-Down or Shortcut Menu" on page 77.

Simultaneously Disable and Mark Menu Item Labels

You can mark and disable menu items at the same time. This is the format:

`[~!. labeltext]`

or

`[!.~ labeltext]`

The ~ is the special character code to disable a menu item and `!.` is the special character code to mark a menu item.

In the example that follows, the Line menu item is disabled and marked with a check mark. As with the previous examples, a DIESEL expression can be used to simultaneously disable and mark a menu item label.

```
[~!.Line]
```

See Also

"DIESEL Expressions in Menu Macros" on page 108

Reference a Pull-Down or Shortcut Menu

Using a method similar to that used to activate regular submenus, you can activate or deactivate a pull-down or pop-up submenu.

The two methods for referencing a pull-down or shortcut menu are relative and absolute. *Relative referencing* uses the menu group and name tag; *absolute referencing* uses the absolute position of the menu item in the menu hierarchy. The relative referencing method is recommended because of its dynamic nature, which allows it to function properly regardless of the current state of the menu.

Relative Referencing of Pull-Down and Shortcut Menu Items

To reference a pull-down or shortcut menu item based on its menu group and name tag, use the AutoLISP **`menucmd`** function. The following syntax references a menu item based on its name tag.

```
(menucmd "Gmenugroup.name_tag=value")
```

The following example disables the menu item `ID_Line` in the `sample` menu group. It works regardless of the menu item's location in the menu.

```
[Disable Line](menucmd "Gsample.ID_Line=~")
```

If the author of a partial menu is aware of the contents of the base menu, the syntax of a menu item can reference a tag from the base file. An excerpt from the base file *acad.mnu* might look something like the following:

```
***MENUGROUP=ACAD
***POP0          (and so forth...)
 ....
***POP6
ID_MnHelp   [Help]
ID_Contents [Contents]^C^C_HELP
ID_About    [About]^C^C_ABOUT
```

A menu item in a partial menu can be modified to have an additional menu item that references the tag in the base menu.

```
***POP2
[Title2]
[Disable Help Contents]^P(menucmd "Gacad.ID_Contents=~") ^P
```

In this manner, multiple partial menu files and specific base files can work together. AutoCAD enforces strict menu group definition so that no two menus can define the same menu group. Attempts to load a menu with a conflicting menu group results in cancellation of the MENULOAD request.

Absolute Referencing of Pull-Down and Shortcut Menu Items

In addition to referencing a menu item, you can activate or deactivate a menu item with the P*n*=*xxx* syntax. This is the format:

```
Pn.i=xxx
```

The P*n* specifies the active POP*n* menu section (0 through 16 are valid values); *i* specifies the menu item number; and *xxx* (if present), specifies a string defining the action.

The following example uses the AutoLISP **menucmd** function to reference a pull-down or shortcut menu item. Because AutoCAD menu files are dynamic (through the loading of partial menus), the following syntax won't work in all cases.

```
[Disable Line Old Way](menucmd "P1.2=~")
```

This syntax relies on the location of the menu item and does not work if a new item was inserted into the POP1 section by the menu author or if a new pull-down menu is inserted before POP1 by the MENULOAD command.

You can use the P*n*=*xxx* syntax from a menu macro if it follows the $ command. The following example disables item 4 in the POP3 section.

```
$P3.4=~
```

The following example adds a check mark to item 1 in the POP7 section.

```
$P7.1=!.
```

The following example removes any disabling or mark character from item 1 in the POP7 section.

```
$P7.1=
```

Menu item numbering is consecutive without regard to the hierarchy of the menu file; item 1 is the first item following the title.

```
***POP5
[Assist  ]                Title
[Help!   ]'?              Item 1
[Cancel  ]^C^C^C          Item 2
[--]                      Item 3
[Undo    ]^C^C_U          Item 4
[Redo    ]^C^C_redo       Item 5
[--]                      Item 6
[->Osnap ]                Item 7
[Center  ]center          Item 8
```

To make it easy for an item to address itself without regard to location in the menu hierarchy, use these forms:

`$P@.@=`*`xxx`*	References the current or most recently chosen menu item.
`$P@.`*`n`*`=`*`xxx`*	References item *`n`* in the current or most recently chosen menu.

AutoLISP Access to Label Status

The AutoLISP **`menucmd`** function accepts `$P`*`n`*`=`*`xxx`* command strings but without the leading `$`. For these functions, the *`xxx`* portion of the command string can have special values.

`P`*`n`*`.`*`i`*`=?`	Returns the current disabled and marked status for the specified item as a string (for example, ~ for a disabled item, `!.` for an item with a check mark, and `""` for an item that is neither grayed out nor marked).
`P`*`n`*`.`*`i`*`=#?`	Returns the same type of string as described for `P`*`n`*`.i=?`, but with the `P`*`n`*`.`*`i`*`=` prefix. This is useful in conjunction with the `@` forms, because the actual menu and item number are returned.

For example, if the fifth item in the `POP6` section is disabled, the following **`menucmd`** code returns the following string values.

```
(menucmd "P6.5=?")      returns  "~"
(menucmd "P6.5=#?")     returns  "P6.5=~"
```

See "Use of AutoLISP in Menu Macros" in the *AutoLISP Developer's Guide*.

Swap and Insert Pull-Down Menus

Because AutoCAD pull-down menus are the cascading type, you usually don't need to swap menus. Also, swapping menus can detract from the consistency of the user interface. However, inserting and removing a pull-down menu can be appropriate when the user specifically loads or unloads an application that requires an additional menu.

Swap Pull-Down Menus

Using $ commands in menu macros, you can swap pull-down menus in specific POP*n* locations. This method, however, is not recommended unless you can verify that the menu you are replacing is really the one you think it is. Because of the dynamic nature of AutoCAD menus, a menu you inserted at position P6 might not actually be at that location. If you try to swap this menu for another, you might not remove the correct menu. An alternative method for menu swapping involves relative (or global) referencing (see "Insert and Remove Pull-Down Menus"). Using this method, you can insert the new menu in front of a known menu and then remove the known menu.

For menu-swapping purposes, the active pull-down menu areas are named P1 through P16. The following menu macro replaces the menu at position P3 with the menu named JoesMenu in the menu group MYMENU.

```
$P3=MyMenu.JoesMenu
```

The same thing can be done with the AutoLISP **menucmd** function as follows:

```
(menucmd "P3=MyMenu.JoesMenu")
```

You can use the $P*n*=* special command from within any menu macro to force the menu currently assigned to area POP*n* to be displayed.

Note The swapping of pull-down menus does not conform to the Microsoft® user interface guidelines and is not guaranteed to be available in future releases of AutoCAD.

Insert and Remove Pull-Down Menus

You can use the AutoLISP **menucmd** function to insert or remove a pull-down menu. The syntax is similar to that used to swap pull-down menus except that the left side of the assignment is the pull-down menu before which you want the new menu to be inserted. The right side of the assignment is a plus sign (+), followed by the name of the menu group, followed by a period and the menu's alias, as shown in the following syntax:

```
(menucmd "Gmenugroup1.menuname1=+menugroup2.menuname2")
```

You can also insert a menu with the P*n*= syntax. The following menu macro inserts a menu after the P5 menu. (You can also use the **menucmd** function with this format.)

```
(menucmd "P5=+mymenu.new3")
```

If you use this method to insert a menu, remember that you cannot rely on its being inserted at the P6 menu location as you might expect. There are two reasons that this may not be the case:

- If the current menu bar has only three menus, inserting a menu after menu P5 results in the new menu's location being P4.
- If the user inserts or removes a menu with the MENULOAD command or when another application inserts or removes menus, menu numbering can get out of sync.

This is the syntax for removing a menu:

```
(menucmd "Gmenugroup.menuname=-")
```

The following example removes the menu NEW3 that is a member of the MyMenu group.

```
(menucmd "Gmymenu.new3=-")
```

As you might expect, the preceding format is preferable to the P*n*= format because it removes only the specified menu. The following example removes the menu at the P4 location (whatever it is).

```
$P4=-
```

Note Use the P*n* syntax as part of the syntax for a **menucmd** statement only. Use the $P*n* syntax for menu macro–specific statements.

Controlling Toolbars Across Partial Menus

To control toolbars across partial menus, use the following syntax at the Toolbar Name prompt of the -TOOLBAR command.

```
menugroup.subsection-name
```

This syntax accesses the toolbar identified by menugroup.menuname and allows you to use the full spectrum of -TOOLBAR command options on that toolbar.

If the menu group is left out of any of these commands and functions, then AutoCAD defaults to the base menu.

You should be aware of the following:

- You cannot swap into the POP0 menu position. However, you can swap a POP0 menu into any other Pop menu position.
- Image menus cannot be swapped from external menu files.
- You can swap menus only of the same type, that is, one Aux for another, one Pop for another, and so on. Trying to swap between types may result in unpredictable behavior. However, within a given type, you can swap any menu for any other menu. This can lead to strange behavior for Tablet menus, because they typically do not all have the same number of macros.

Customize Toolbars

The Toolbars section of the MNU file specifies the default layout and contents of the toolbars. It contains a submenu for each toolbar defined by the menu.

Create Toolbars

In the Toolbars section of an MNU file, you can create toolbars with buttons, flyouts, and special control elements and use your own bitmaps for the button icons.

If you just want to create or change toolbars or create, rearrange, add, or remove buttons and flyouts, you can use CUSTOMIZE. See "Create Custom Toolbars".

The menu item syntax for the Toolbars section of the MNU file is shown in the following example. All lines other than the separator begin with a standard name tag, which is used to associate help information with the item. In the example, **TOOLS1 is a submenu that uses the alias TOOLS1 as a label to reference the subsequent toolbar definition.

```
***TOOLBARS
**TOOLS1
TAG1 [Toolbar ("tbarname", orient, visible, xval, yval, rows)]
TAG2 [Button ("btnname", id_small, id_large)]macro
TAG3 [Flyout ("flyname", id_small, id_large, icon, alias)]macro
TAG4 [Control (element)]
[--]
```

The first line of a toolbar submenu is the toolbar definition (TAG1 in the example), which defines the characteristics of the toolbar. The remaining lines in the submenu can be a mix of the remaining toolbar items. The second line in the example above (TAG2) defines a button. The third line (TAG3) defines a flyout control, and the fourth line (TAG4) defines a special control element. The fifth line defines a separator (--).

The toolbar definition includes the keyword Toolbar and a series of options that are contained in parentheses. The options define the display characteristics of the toolbar.

```
TAG1 [Toolbar ("tbarname", orient, visible, xval, yval, rows)]
```

The options are as follows:

tbarname	The string that names the toolbar. The string must include alphanumeric characters with no punctuation other than a dash (–) or an underscore (_). With this name (along with the alias) the toolbar can be referenced programmatically.
orient	The orientation of the toolbar. The values are `floating`, `top`, `bottom`, `left`, and `right` and are not case-sensitive.
visible	The visibility of the toolbar. The values, `show` and `hide`, are not case-sensitive.
xval	A numeric value specifying the *X* coordinate in pixels. This value is measured from the left edge of the screen to the left side of the toolbar.
yval	A numeric value specifying the *Y* coordinate in pixels. This value is measured from the top edge of the screen to the top of the toolbar.
rows	A numeric value specifying the number of rows.

The folllowing example is the first few lines of the Zoom toolbar in *acad.mnu*:

```
**TB_ZOOM
ID_TbZoom [_Toolbar("Zoom", _Floating, _Hide, 100, 380, 1)]
ID_ZoomWindo [Button("Zoom Window", ICON_16_ZOOWIN,
ICON_16_ZOOWIN,)]'_zoom_w
ID_ZoomDynam [Button("Zoom Dynamic", ICON_16_ZOODYN,
ICON_16_ZOODYN,)]'_zoom_d
ID_ZoomScale [Button("Zoom Scale", ICON_16_ZOOSCA,
ICON_16_ZOOSCA,)]'_zoom_s
```

Note Each line begins with ID and there are no line breaks.

To control Toolbars with partial menus, use the following syntax at the Toolbar Name prompt of the -TOOLBAR command:

menugroup.toolbarname

The following AutoLISP code displays the toolbar MYBAR in the menu group MYGROUP. (This code assumes that the MYGROUP menu is already loaded.)

```
(command "toolbar" "mygroup.mybar" "show")
```

If *menugroup* is not included, then AutoCAD defaults to the base menu.

See Also

"Create Toolbar Buttons" on page 84
"Create Toolbar Flyouts" on page 85
"Define Controls for a Toolbar" on page 86
"Specify User-Defined Bitmaps" on page 87

Create Toolbar Buttons

Toolbar button definitions in the MNU file contain the keyword Button followed by options that are contained in parentheses.

```
TAG2 [Button ("btnname", id_small, id_large)]macro
```

The options are as follows:

btnname	The string that names the button. The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). This string is displayed as a tooltip when the cursor is placed over the button.
id_small	The string that names the ID string of the small-image resource (16 × 15 bitmap). The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). It can also specify a user-defined bitmap (see "Specify User-Defined Bitmaps" on page 87).
id_big	The string that names the ID string of the large-image resource (32 × 30 bitmap). If the specified bitmap is not 32 × 30, AutoCAD scales it to that size. The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). This can also specify a user-defined bitmap (see "Specify User-Defined Bitmaps" on page 87).

macro — The menu item macro. It follows the standard menu macro syntax.

See Also

"Create Toolbars" on page 82

Create Toolbar Flyouts

Toolbar flyout definitions in the MNU file contain the keyword `Flyout` followed by options that are contained in parentheses.

`TAG3 [Flyout ("`*`flyname`*`", `*`id_small`*`, `*`id_large`*`, `*`icon`*`, `*`alias`*`)]`*`macro`*

The options are as follows:

flyname — The string that names the flyout. The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). This string is displayed as a tooltip when the cursor is placed over the flyout.

id_small — The string that names the ID string of the small-image resource (16 × 15 bitmap). The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). This can also specify a user-defined bitmap (see "Specify User-Defined Bitmaps" on page 87.

id_big — The string that names the ID string of the large-image resource (32 × 30 bitmap). If the specified bitmap is not 32 × 30, AutoCAD scales it to that size. The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). It can also specify a user-defined bitmap (see "Specify User-Defined Bitmaps" on page 87.

icon — The keyword that controls whether to display either its own icon or the last icon selected (other). The acceptable values, `OwnIcon` and `OtherIcon`, are not case-sensitive.

alias — The reference to the toolbar to display as the flyout. The alias refers to a toolbar submenu defined with the standard `**`*`aliasname`* syntax.

macro — The menu item macro. It follows the standard menu macro syntax.

See Also

"Create Toolbars" on page 82

Define Controls for a Toolbar

Toolbar control definitions in the MNU file contain the keyword `Control` followed by a name specifying the type of control element you want contained in parentheses. For examples of the Color, Linetype, and Lineweight controls, see the Properties toolbar in AutoCAD.

```
TAG4 [Control (element)]
```

The values for the *element* parameter specify the following controls (the values are not case-sensitive):

_Color	Color control element. This element is a drop-down list that provides specification of the current color.
_Dimstyle	Dimension style control element. This element is a drop-down list that provides specification of the current dimension style.
_Layer	Layer control element. This element is a drop-down list that provides control of the current layers in the drawing.
_Linetype	Linetype control element. This element is a drop-down list that provides specification of the current linetype.
_Lineweight	Lineweight control element. This element is a drop-down list that provides specification of the current lineweight.
_PlotStyle	Plot style control element. This element is a drop-down list that provides specification of the current plot style.
_Refblkname	Xref name control element. This element displays the current xref name in edit mode.
_UCSManager	UCS control element. This element is a drop-down list that provides specification of the current UCS.
_View	View control element. This element is a drop-down list that provides specification of the current standard 3D views.
_ViewportScale	Viewport scale control element. This element is a drop-down list that provides specification of viewport scaling in layouts.

See Also

"Create Toolbars" on page 82

Specify User-Defined Bitmaps

User-defined bitmaps can be used in place of the *id_small* and *id_big* image resource names in the button and flyout menu items.

A user-defined bitmap must be of the proper size for the *id_small* parameter (16 pixels wide by 15 pixels high) and must reside in the library search path. For the *id_big* parameter, if the specified bitmap is not 32 × 30, AutoCAD scales it to that size. Specify a user-defined bitmap with the file name and *.bmp* extension as shown in the following example:

```
TAG34 [Button ("My Command", mycmd16.bmp, mycmd32.bmp)]^C^CMYCMD
```

See Also

"Create Toolbars" on page 82

Create Image Tile Menus

The main purpose of an image tile menu is to provide an image when the user must select a graphical symbol instead of text.

Overview of Image Tile Menus

You define an image tile menu by including an Image section in the menu file. AutoCAD displays images in groups of 20, along with a scrolling list box containing the associated slide file names or related text. Image tile submenus are unlimited in length. If an image tile submenu contains more than 20 slides, AutoCAD provides Next and Previous buttons so that the user can leaf through pages of images.

The Image section uses submenus similar to the Toolbars and Screen sections. As with Pop menu sections, the first line of the submenu is its title. The title is displayed as the label of the dialog box that contains the images. Submenus should be separated by at least one blank line to clear out items from a previous submenu.

Image tile menu items use item labels to define the text of the scrolling list and the image itself. The label is followed by an associated menu macro. Image tile menus cannot contain name tags.

Image Tile Item Labels

Labels in an image tile menu generally refer to slide file names instead of text labels that are displayed on the screen. The slide file contains the image to show for that selection. The name of the slide file, which can be a single slide or part of a library, should appear exactly as you would enter it at the VSLIDE command.

When successive slides from the same library are displayed, the library file remains open. Therefore, the time required to display an image menu is significantly reduced. The SLIDELIB utility can be used to combine multiple slide files into a slide library.

Image tile menu labels are displayed in a scrolling list box that can accommodate up to 19 characters per label. The slide file name is typically displayed; however, the following icon menu-labeling options are also available.

`[`*`sldname`*`]` The slide name *sldname* is displayed in the list box, and the slide *sldname* is displayed as an image.

`[`*`sldname`*`,`*`labeltext`*`]` The text *labeltext* is displayed in the list box, and the slide *sldname* is displayed as an image.

`[`*`sldlib`*`(`*`sldname`*`)]` The slide name *sldname* is displayed in the list box, and the slide *sldname* in the slide library *sldlib* is displayed as an image.

`[`*`sldlib`*`(`*`sldname`*`,`*`labeltext`*`)]` The text *labeltext* is displayed in the list box, and the slide *sldname* in the slide library *sldlib* is displayed as an image.

`[blank]` When you supply the text `blank` as an icon label, a separator line is displayed in the list box and a blank image is displayed.

`[ `*`labeltext`*`]` When the first character of an item label is a space, the text supplied as *labeltext* is displayed in the list box and no image is displayed. In this case you can include related commands and simple items such as Exit without needing to make slides that contain those words.

Image Tile Menu Macros

Image tile menu macros can perform the same function as other menu macros; however, you cannot use the menu macro repetition feature. These menu macros can contain menu commands, including $I= commands. It is possible, therefore, to construct hierarchical image tile menus in which a selection displays another image tile menu, and so on. Because the activation of these menus is sequential rather than nested, the complexity of the structures you can create has no limits.

Display of Image Tile Menus

The $I= macro command calls the image tile menu. Before you can display an image tile menu you must load it using the following syntax:

```
$I=[menugroup.]menuname
```

The $I=* macro command displays the currently loaded image tile menu. For example, the following macros load and display the IMAGE_POLY image tile menu in *acad.mnu*.

```
$I=image_poly $I=*
```

The following example loads and displays the MYBLOCKS image menu from a partially loaded menu group MYGROUP.

```
$I=mygroup.myblocks $I=*
```

You can also use the AutoLISP **menucmd** function to load and display image tile menus. The following code produces the same result as the previous example.

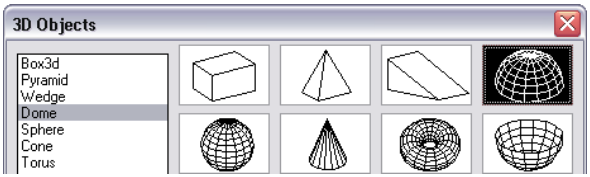
```
(menucmd "I=mygroup.myblocks")
(menucmd "I=*")
```

Sample Image Tile Menus

This example shows an image tile submenu named 3DObjects.

```
**3DOBJECTS
[3D Objects]
[acad(box3d)]^c^cai_box
[acad(Pyramid)]^c^cai_pyramid
[acad(Wedge)]^c^cai_wedge
[acad(Dome)]^c^cai_dome
[acad(Sphere)]^c^cai_sphere
[acad(Cone)]^c^cai_cone
[acad(Torus)]^c^cai_torus
[acad(Dish)]^c^cai_dish
[acad(Mesh)]^c^cai_mesh
```

The resulting figure is a portion of the image tile menu.



3D Objects image tile menu sample

In the next example, an image tile menu is used to insert various electronic parts. The text label is an item that swaps to another image tile submenu that contains various fasteners.

```
***IMAGE
**IPARTS
[Electronic Parts]
[cap]^Cinsert cap
[res]^Cinsert res
[neon]^Cinsert neon
[triode]^Cinsert triode
[tetrode]^Cinsert tetrode
[ Fasteners]$I=ifast $I=*

**IFAST
[Fasteners]
[nut632]...
```

To activate this image tile menu, you could choose a menu item such as the following from any menu.

```
[Electronic parts]$I=iparts $I=*
```

In the following variation, the images are retrieved from a slide library named *elib*; only the slide name appears in the list box.

```
***IMAGE
**IPARTS
[Electronic Parts]
[elib(cap)]^Cinsert cap
[elib(res)]^Cinsert res
[elib(neon)]^Cinsert neon
[elib(triode)]^Cinsert triode
[elib(tetrode)]^Cinsert tetrode
```

Prepare Slides for Image Tile Menus

You can use any slide generated by AutoCAD as an image. However, the optimal use of image tile menus requires that you take care in preparing slides that will serve as images. Keep the following suggestions in mind as you prepare slides for an image tile menu.

- Keep the image simple. When an image tile menu is displayed, the user must wait for all images to be drawn before making a selection. If you are showing the user numerous complex symbols, make the images simple versions rather than full renditions. An image should be as simple as possible and yet immediately recognizable.
- Fill the box. Screen space is limited, and images appear in small portions of the full screen. When making a slide for an image, be sure to fill the screen with the image before starting MSLIDE. If the image is very wide and short, or long and thin, the image tile menu will look best if you use PAN to center the image on the screen before making the slide.

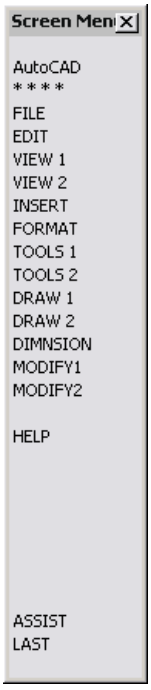
  Images are displayed with an aspect ratio of 3:2 (3 units wide by 2 units high). If your drawing area has a different aspect ratio, it can be difficult to produce image slides that are centered in the image tile menu. If you work within a layout viewport that has an aspect ratio of 3:2, you can position the image and be assured that it will look the same when it is displayed in the image tile menu.
- Use the -SHADEMODE command for solid-filled areas before you generate the slide. Otherwise, objects such as wide polylines and traces are displayed as outlines.
- Remember the purpose of these images. Do not make use of images to encode abstract concepts into symbols. Image tiles are useful primarily when the user must select a graphic symbol.

Create Screen Menus

The screen menu section of the MNU file controls the screen menu area.

By default, the screen menu is disabled. To enable the screen menu, click Display Screen Menu on the Display tab in the Options dialog box.

In the MNU file, the `***SCREEN` section label represents the beginning of the AutoCAD screen menus. The submenu section label shown here is identified by the string `**S`. A simple, concise name, such as this, is convenient when many separate items reference this submenu, as shown in the following example:

Screen menu	Screen menu file section
	***SCREEN **S [AutoCAD]^C^C^P(ai_rootmenus) ^P [* * * *]$S=ACAD.OSNAP [FILE]$S=ACAD.01_FILE [EDIT]$S=ACAD.02_EDIT [VIEW 1]$S=ACAD.03_VIEW1 [VIEW 2]$S=ACAD.04_VIEW2 [INSERT]$S=ACAD.05_INSERT [FORMAT]$S=ACAD.06_FORMAT [TOOLS 1]$S=ACAD.07_TOOLS1 [TOOLS 2]$S=ACAD.08_TOOLS2 [DRAW 1]$S=ACAD.09_DRAW1 [DRAW 2]$S=ACAD.10_DRAW2 [DIMNSION]$S=ACAD.11_DIMENSION [MODIFY1]$S=ACAD.12_MODIFY1 [MODIFY2]$S=ACAD.13_MODIFY2 [HELP]$S=ACAD.14_HELP [ASSIST]$S=ACAD.ASSIST [LAST]$S=ACAD.

Screen Submenus

Screen menu submenu labels have the following format:

`**`*`menuname`* `[`*`startnum`*`]`

The *menuname* is a string of up to 33 characters containing letters, digits, and the dollar ($), hyphen (-), and underscore (_) characters. The submenu label must reside on a menu file line by itself and must not contain embedded blanks. An optional integer *startnum*, which specifies the start line of the submenu, can follow *menuname*.

A submenu can contain any number of items, but the total size of the screen menu is limited by the setting of the SCREENBOXES system variable (typically set to 28). For instance, if a screen menu submenu has 21 items, but the screen can display only 20 items at a time, the last item in the submenu is inaccessible.

When a submenu is activated, its items normally replace those of the previous menu starting at the beginning (menu box 1) and continuing through all items of the submenu. Thus, a submenu can replace only a portion of the previous menu. You can add an item number after the section or submenu label to specify a replacement starting with a menu item other than 1, as shown in the following example:

```
**SAMPLE 3
```

When the SAMPLE submenu is activated, the first two menu boxes are unchanged and submenu replacement begins with menu box 3.

To restore the previous screen items, a menu item must issue the following code without a submenu label.

```
$S=
```

AutoCAD keeps track of the last eight submenus. If you exceed eight, the first menus are discarded.

The following sample screen menu section demonstrates the use of submenus.

```
***SCREEN
[EASYmenu]
```
Blank line

```
[DRAW... ]$S=Draw_Root
[EDIT... ]$S=Edit_Root
```
Blank line

```
[Bye     ]end
```
Three blank lines fill out this page of the menu to 10 lines and blank out items displayed by the submenus. Because no submenu extends below this line, it is displayed in all menus. It recalls the main menu.

```
[ -MAIN- ]$S=SCREEN
```

```
**Draw_Root 2
```
The 2 after the submenu name starts this menu on the line after `[EASYmenu]`.

```
[Line    ]line
[Circle  ]circle
[Arc     ]arc
```
At least one blank line.

```
**Edit_Root 2

[Erase ]$S=Sel_obj erase
[Copy  ]$S=Sel_obj copy
[Move  ]$S=Obj_sel move
```
Notice the use of a menu alias. At least two blank lines cover up the `Sel_obj` *menu items.*

```
**Obj_sel 2                    You can use both Obj_sel
**Sel_obj 2                    and Sel_obj to call this menu.
[Last    ]last
[Previous]previous
[Window  ]window
[Crossing]crossing

[ -PREV- ]$S=                  The $S= calls the previous menu.

***BUTTONS1                    Pointing-device button menu.
;                              Assigns ENTER to button 2.
redraw                         Assigns REDRAW command to
                               button 3.
```

The previous example contains three submenus: `Draw_Root`, `Edit_Root`, and `Sel_obj`.

`Draw_Root` and `Edit_Root` are called from the main screen menu when you select the Draw or Edit menu items. The `Draw_Root` submenu provides three selection items that correspond to AutoCAD commands. The `Edit_Root` submenu also contains three selection items, each of which calls the submenu `Sel_obj` before executing the appropriate command.

In all cases, a `–MAIN–` selection item recalls the main screen menu. A screen menu writes over (erases) only as many lines of the previous screen menu as it contains. If a screen menu contains more items than boxes on the screen, or if a buttons menu contains more items than buttons available, the excess items are ignored.

You can use blank lines in menu files to lengthen submenus so that they cover up previous menus. You can also include blank lines to improve the readability of the file.

Selecting a menu item called Zoom from the main screen menu can activate a submenu containing the options for the ZOOM command.

For an alternative method of calling a command submenu, see "Automatic Swapping of Screen Submenus" on page 96.

The following example references the submenu `**01_FILE` in the `ACAD` menu group.

```
[FILE    ]$S=ACAD.01_FILE
```

Most screen menus in *acad.mnu* are loaded at menu box 3, enabling the menu labels `[AutoCAD]` and `[* * * *]` to remain on the screen.

The following example shows how the **01_FILE submenu is displayed on the screen. Notice that the first line (for New) is displayed at menu box 3.

Screen menu	Screen menu file section
	**01_FILE 3 [New]^C^C_new [Open]^C^C_open [Qsave]^C^C_qsave [Saveas]^C^C_saveas [Export]^C^C_export [Config]^C^C_config [Plot]^C^C_plot [Audit]^C^C_audit [Recover]^C^C_recover [Purge]^C^C_purge [Quit]^C^C_quit

The menu items Assist and Last are displayed at the bottom of the screen menu area because they are part of the **s submenu that is not overwritten by the **01_FILE submenu.

Screen Menu Item Labels

If a screen menu item does not contain an item label, the first eight characters of a menu macro appear on the screen menu. The command in the following example would be displayed as SNAP 0.0.

```
SNAP 0.001
```

If a label is provided, the first eight characters of the label are displayed in the appropriate screen menu box. Any additional characters can serve as comments.

Note The maximum number of menu items depends on your system. You can retrieve the number of screen menu boxes with the SCREENBOXES system variable.

Automatic Swapping of Screen Submenus

The MENUCTL system variable controls the automatic swapping of screen submenus when a corresponding command is issued. When MENUCTL is set to 1 (on) and an AutoCAD command is called from a menu item, AutoCAD issues a `$S=cmdname` (where *cmdname* is the name of the command), which calls a screen submenu of the same name as the command. The Standard menu, *acad.mnu,* takes advantage of this feature by setting MENUCTL to 1 from the *acad.mnl* file. Setting MENUCTL to 0 (off) affects the operation of the Standard menu but may be preferable for older custom menus.

To display the screen menu

1. On the Tools menu, click Options.
2. Click the Display tab.
3. On the Display tab under Window Elements, select Display Screen Menu.
4. Click OK.

Create Tablet Menus

You can configure up to four areas of your digitizing tablet as menu areas for command input. The sections of the menu file labeled `TABLET1` through `TABLET4` define the menu macros associated with tablet selections in these areas.

The menu items in Tablet sections use the same syntax as those in the other sections. Item labels can be used as comments and are not displayed.

The tablet menu areas that you define with the Cfg option of the TABLET command are divided into equal-sized menu selection boxes, which are determined by the number of columns and rows you specify in each area. These tablet menu selection boxes correspond directly to the lines that follow the Tablet section labels in a left-to-right, top-to-bottom order (whether or not they contain text).

For example, if you configure a menu area for five columns and four rows, the menu item on the line immediately following the section label corresponds to the leftmost selection box in the top row. Similarly, the menu item on the eighth line following the section label corresponds to the third box from the left in the second row. AutoCAD can recognize up to 32,766 menu items in each tablet section, which should be more than enough for any tablet menu.

You can add your own menu macros to the `***TABLET1` section of *acad.mnu*. The menu item labels in this area correspond to the 225 boxes at the top of your tablet template (rows A through I and columns 1 through 25). You can add your menu macro after the corresponding *`[row-column]`* menu label, using standard menu item syntax.

```
***TABLET1
[A-1]
[A-2]
[A-3]
.
.
.
[I-25]
```

It is not recommended that you modify any lines following box I-25.

See Also

"Create Menu Macros" on page 54

Create Status Line Help Messages

Status line help messages are the simple, descriptive messages that are displayed on the status line when a menu item is chosen. The Helpstrings menu section defines these messages.

The following example shows a simple menu file that makes use of the Helpstrings menu section.

```
***MENUGROUP=sample
***POP1
ID_Title  [/TTitle]
ID_Cancel [Cancel Command]^C^C
ID_Line   [/LLine]^C^C_line
          [Disable Line](menucmd "Gsample.ID_Line=~")
          [Check Line](menucmd "Gsample.ID_Line=!.")

***POP2
[/2Title2]
[Another Pull Down](menucmd "Gsample.ID_Line=~")

***HELPSTRINGS
ID_Title  [This is the Title menu]
ID_Cancel [This item cancels the previous command]
ID_Line   [This draws a simple line]
```

The syntax for the Helpstrings section is a name tag followed by a label. When a menu item is highlighted, the name tag for that item is queried for a corresponding entry in the Helpstrings section. If a match occurs, the string contained within the label is displayed on the status line.

Create Shortcut Keys

You can define your own shortcut keys (or accelerator keys). The following is a short example of an Accelerators section in an MNU file.

```
***ACCELERATORS
ID_Line    [SHIFT+CONTROL+"L"]
[CONTROL+"Q"]^C^C_quit
[CONTROL+SHIFT+"Z"]^C^Czoom extents
```

The Accelerators section contains menu items in one of two formats, both illustrated in the above example:

- The first menu item maps a key sequence to a menu item: `ID_Line [SHIFT+CONTROL+"L"]`. A name tag (`ID_Line`) is followed by a label containing modifiers: `SHIFT+CONTROL+`. The modifiers are followed by either a single-character key string (`L` in the example) or a special virtual key string (such as `F12`) enclosed in quotation marks. When a special key sequence is recognized, the menu item associated with the name tag is executed as if the user had chosen the menu item.
- The second and third menu items map a key sequence to a menu macro, not a menu item. A label containing a modifier and key string (`CONTROL+"Q"`) is followed by a menu macro (`^C^C_quit `). The menu macro uses the standard format and special characters, except that the backslash character (\) cannot be used to pause for user input.

If you want a pause for user input, use the first method to map a key sequence to a menu item that includes the desired pause.

You can concatenate modifiers by using the plus symbol (+), as in the first and third examples. The valid modifiers are CONTROL (the CTRL key) and SHIFT (the SHIFT key).

The following table lists the special virtual keys that you can use with modifiers. Both standard keyboard letters and numbers and these special virtual keys must be enclosed in quotation marks.

Special virtual keys

String	Description	Exceptions
F1	F1 key	It is not recommended that you assign a menu macro to the F1 key, because F1 is generally associated with Help. Using a modifier with this key is acceptable.
F2	F2 key	Unmodified, switches the state of the text window.
F3	F3 key	Unmodified, runs OSNAP.
F4	F4 key	Unmodified, turns TABMODE on or off.
F5	F5 key	Unmodified, turns ISOPLANE on or off.
F6	F6 key	Unmodified, turns COORDS on or off.
F7	F7 key	Unmodified, turns GRIDMODE on or off.
F8	F8 key	Unmodified, turns ORTHOMODE on or off.
F9	F9 key	Unmodified, turns SNAPMODE on or off.
F10	F10 key	Unmodified, turns Polar Tracking on or off.
F11	F11 key	Unmodified, turns Object Snap Tracking on or off.
F12	F12 key	None
INSERT	INS key	Must be used with the CONTROL modifier.

Special virtual keys (*continued*)

String	Description	Exceptions
DELETE	DEL key	Must be used with the CONTROL modifier.
ESCAPE	ESC key	It is not recommended that you assign a menu macro to the ESC key, because ESC is generally associated with Cancel. CONTROL+ESCAPE and CONTROL+SHIFT+ESCAPE cannot be assigned a menu macro; these sequences are controlled by Windows. Using the SHIFT modifier with this key is acceptable.
UP	UPARROW key	Must be used with the CONTROL modifier.
DOWN	DOWNARROW key	Must be used with the CONTROL modifier.
LEFT	LEFTARROW key	Must be used with the CONTROL modifier.
RIGHT	RIGHTARROW key	Must be used with the CONTROL modifier.
NUMPAD0	0 key	None
NUMPAD1	1 key	None
NUMPAD2	2 key	None
NUMPAD3	3 key	None
NUMPAD4	4 key	None
NUMPAD5	5 key	None
NUMPAD6	6 key	None
NUMPAD7	7 key	None
NUMPAD8	8 key	None
NUMPAD9	9 key	None

Coordinate Entry with Shortcut Keys (Example)

Users who make extensive use of coordinate entry might find the following menu enhancement very useful.

```
["NUMPAD5"]@x^h
["NUMPAD6"]<0
["NUMPAD9"]<45
["NUMPAD8"]<90
["NUMPAD7"]<135
["NUMPAD4"]<180
["NUMPAD1"]<-135
["NUMPAD2"]<-90
["NUMPAD3"]<-45
```

If this code is added to the Accelerators section, your numeric keypad is modified as follows: the ENTER key enters the **@** symbol and the other number keys enter the less-than symbol (<), followed by the angular value represented by its location in the keypad. For example, if you wanted to draw a square that was 3 units on each side, you would enter the following:

Command: **line**
From point: *(specify start point)*
To point: *(press number 5)* **3** *(press number 6)*
To point: *(press number 5)* **3** *(press number 2)*
To point: *(press number 5)* **3** *(press number 4)*
To point: **c**