Chapter

III-13

Procedure Windows

Overview 350
Types of Procedure Files 350
Working with the Built-in Procedure Window 350
    Compiling the Procedures 351
    Templates Pop-Up Menu 351
    Procedure Navigation Bar 351
    Write-Protect Icon 351
    Magnifier Icon 351
Creating Procedure Windows 351
    Creating New Procedure Files 352
    Opening an Auxiliary Procedure File 352
Showing Procedure Windows 352
Hiding and Killing Procedure Windows 353
Global Procedure Files 353
    Saving Global Procedure Files 354
Shared Procedure Files 354
    Saving Shared Procedure Files 354
Including a Procedure File 354
Creating Packages 355
Invisible Procedure Files 355
    Invisible Procedure Windows Using #pragma hide 355
    Invisible Procedure Windows Using Independent Modules 356
    Invisible Procedure Files Using The Files Visibility Property 356
Inserting Text Into a Procedure Window 357
Adopting a Procedure File 357
Auto-Compiling 358
Debugging Procedures 358
Finding Text 358
Replacing Text 358
Printing Procedure Text 358
Indentation 359
Document Settings 359
Text Format Settings 359
Syntax Coloring 359
Procedure Window Preferences 360
Double and Triple-Clicking 360
Matching Characters 360
Code Comments 361
Procedure File Text Encodings 361
Procedure Window Shortcuts 362

Overview

This chapter explains what procedure windows are, how they are created and organized, and how you work with them. It does not cover programming. See Chapter IV-2, **Programming Overview** for an introduction.

A procedure window is where Igor procedures are stored. Igor procedures are the macros, functions and menu definitions that you create or that Igor creates automatically for you.

The content of a procedure window is stored in a procedure file. In the case of a packed Igor experiment, the procedure file is packed into the experiment file.

Types of Procedure Files

There are four types of procedure files:

- The experiment procedure file, displayed in the built-in procedure window
- Global procedure files, displayed in auxiliary procedure windows
- Shared procedure files, displayed in auxiliary procedure windows
- Auxiliary experiment procedure files, displayed in auxiliary procedure windows

The built-in procedure window holds experiment-specific procedures of the currently open experiment. This is the only procedure window that beginning or casual Igor users may need.

All other procedure windows are called **auxiliary** to distinguish them from the built-in procedure window. You create an auxiliary procedure window using Windows→New→Procedure. You can then save it to a standalone file, using File→Save Procedure As, or allow Igor to save it as part of the current experiment.

A global procedure file contains procedures that you might want to use in any experiment. It must be saved as a standalone file in the "Igor Procedures" folder of your **Igor Pro User Files** folder. Procedure files in "Igor Procedures" are automatically opened by Igor at startup and left open until Igor quits. This is the easiest way to make procedures available to multiple experiments.

A shared procedure file contains procedures that you want to use in more than one experiment but that you don't want to be open all of the time. It must be saved as a standalone file. The recommended location is the "User Procedures" folder of your **Igor Pro User Files** folder.

An auxiliary experiment procedure file contains procedures that you want to use in a single experiment but want to keep separate from the built-in procedure window for organizational purposes. In a packed experiment it is saved as a packed file within the experiment file. In an unpacked experiment it is saved as a standalone file in the experiment folder.

Working with the Built-in Procedure Window

Procedures that are specific to the current experiment are usually stored in the built-in procedure window. Also, when Igor automatically generates procedures, it stores them in the built-in procedure window.

To show the built-in procedure window, choose Windows→Procedure Windows→Procedure Window or press Command-M (*Macintosh*) or Ctrl+M (*Windows*). To hide it, click the close button or press Command-W (*Macintosh*) or Ctrl+W (*Windows*).

To create a procedure, just type it into the procedure window.

The contents of the built-in procedure window are automatically stored when you save the current Igor experiment. For unpacked experiments, the contents are stored in a file called “procedure” in the experiment folder. For packed experiments, the contents are stored in the packed experiment file. When you open an experiment Igor loads its procedures back into the built-in procedure window.

Compiling the Procedures

When you modify the text in the procedure window, you will notice that a Compile button appears at the bottom of the window.

Clicking the Compile button scans the procedure window looking for macros, functions and menu definitions. Igor compiles user-defined functions, generating low-level instructions that can be executed quickly.

Igor also compiles the code in the procedure window if you choose Compile from the Macros menu or if you activate any window other than a procedure or help window.

Templates Pop-Up Menu

The Templates pop-up menu lists all of the built-in and external operations and functions in alphabetical order and also lists the common flow control structures.

If you choose an item from the menu, Igor inserts the corresponding template in the procedure window.

If you select, click in, or click after a recognized operation, function or flow-control keyword in the procedure window, two additional items are listed at the top of the menu. The first item inserts a template and the second takes you to help.

Procedure Navigation Bar

The procedure navigation bar appears at the top of each procedure window. It consists of a menu and a settings button. The menu provides a quick way to find procedures in the active procedure window.

Certain syntax errors prevent Igor from parsing procedures. In that case, the menu shows no procedures.

You can hide the navigation bar by choosing Misc-Miscellaneous Settings, selecting the Text Editing section, and unchecking the Show Navigation Bar checkbox.

Write-Protect Icon

Procedure windows have a write-enable/write-protect icon which appears in the lower-left corner of the window and resembles a pencil. If you click this icon, Igor Pro displays an icon indicating that the procedure window is write-protected. The main purpose of this is to prevent accidental alteration of shared procedure files.

Igor opens included user procedure files for writing but turns on the write-protect icon so that you will get a warning if you attempt to change them. If you do want to change them, simply click the write-protect icon to turn protection off. You can turn this default write-protection off via the Text Encoding section of the Miscellaneous Settings dialog.

If a procedure file is opened for reading only, you will see a lock icon instead of the pencil icon. A file opened for read-only can not be modified.

WaveMetrics procedures, in the WaveMetrics Procedures folder, are assumed to be the same for all Igor users and should not be modified by users. Therefore, Igor opens included WaveMetrics procedures for reading only.

Magnifier Icon

You can magnify procedure text to make it more readable. See **Text Magnification** on page II-47 for details.

Creating Procedure Windows

There are three ways to create procedures:

- Automatically by Igor
- Manually, when you type in a procedure window
- Semiautomatically, when you use various dialogs

Igor offers to automatically create a **window recreation macro** when you close a target window. A window recreation macro is a procedure that can recreate a graph, table, page layout or control panel window. See **Saving a Window as a Recreation Macro** on page II-42 for details.

You can add procedures by merely typing them in a procedure window.

You can create user-defined controls in a graph or control panel. Each control has an optional **action procedure** that runs when the control is used. You can create a control and its corresponding action procedure using dialogs that you access through the Add Controls submenu in the Graph or Panel menus. These action procedures are initially stored in the built-in procedure window.

You can create user-defined curve fitting functions via the Curve Fitting dialog. These functions are initially stored in the built-in procedure window.

Creating New Procedure Files

You create a new procedure file if you want to write procedures to be used in more than one experiment.

Note: There is a risk in sharing procedure files among experiments. If you copy the experiment to another computer and forget to also copy the shared files, the experiment will not work on the other computer. See **References to Files and Folders** on page II-22 for further details.

If you do create a shared procedure file then you are responsible for copying the shared file when you copy an experiment that relies on it.

To create a new procedure file, choose Windows→New→Procedure. This creates a new procedure *window*. The procedure *file* is not created until you save the procedure window or save the experiment.

You can explicitly save the procedure window by choosing File→Save Procedure As or by closing it and choosing to save it in the resulting dialog. This saves the file as an auxiliary procedure file, separate from the experiment.

If you don't save the procedure window explicitly, Igor saves it as part of the current experiment the next time you save the experiment.

Opening an Auxiliary Procedure File

You can open a procedure file using the File→Open File→Procedure menu item.

When you open a procedure file, Igor displays it in a new procedure window. The procedures in the window can be used in the current experiment. When you save the current experiment, Igor will save a *reference* to the shared procedure file in the experiment file. When you later open the experiment, Igor will reopen the procedure file.

For procedure files that you use from a large number of experiments, it is better to configure the files as global procedure files. See **Global Procedure Files** on page III-353.

For commonly used auxiliary files, it is better to use the include statement. See **Including a Procedure File** on page III-354.

Showing Procedure Windows

We usually show procedure windows when we are doing programming and hide them for normal use.

To show the built-in procedure window, choose Windows→Procedure Windows→Procedure Window or press Command-M (*Macintosh*) or Ctrl+M (*Windows*). To show auxiliary procedure windows, use the Windows→Procedure Windows submenu.

If you have more than one procedure window, you can cycle to the next procedure window by pressing Command-Option-M (*Macintosh*) or Ctrl+Alt+M (*Windows*). Pressing Command-Shift-Option-M (*Macintosh*) or Ctrl+Shift+Alt+M (*Windows*) hides the active procedure window and shows the next one.

You can also show a procedure window by choosing a menu item added by that window while pressing Option (*Macintosh*) or Alt (*Windows*). This feature works only if the top window is a procedure window.

You can show all procedure windows and hide all procedure windows using the Windows→Show and Windows→Hide submenus.

Hiding and Killing Procedure Windows

The built-in procedure window always exists as part of the current experiment. You can hide it by clicking the close button, pressing Command-W (*Macintosh*) or Ctrl+W (*Windows*) or by choosing Hide from the Windows menu. You can not kill it.

Auxiliary procedure files can be opened (added to the experiment), hidden and killed (removed from the experiment). This leads to a difference in behavior between auxiliary procedure windows and the built-in procedure window.

When you click the close button of an auxiliary procedure file, Igor presents the Close Procedure Window dialog to find out what you want to do.

If you just want to hide the window, you can press Shift while clicking the close button. This skips the dialog and just hides the window.

Killing a procedure window closes the window and removes it from the current experiment but does not delete or otherwise affect the procedure file with which the window was associated. If you have made changes or the procedure was not previously saved, you will be presented with the choice of saving the file before killing the procedure.

The Close item of the Windows menu and the equivalent, Command-W (*Macintosh*) or Ctrl+W (*Windows*), behave the same as the close button.

Global Procedure Files

Global procedure files contain procedures that you want to be available in all experiments. They differ from other procedure files in that Igor opens them automatically and never closes them. Configuring a procedure file as a global procedure file is the easiest way to make it widely available.

When Igor starts running, it searches "Igor Pro 7 Folder/Igor Procedures" and "Igor Pro User Files/Igor Procedures" (see **Igor Pro User Files** on page II-31 for details), as well as files and folders referenced by aliases or shortcuts. Igor opens any procedure file that it finds during this search as a global procedure file.

You should save your global procedure files in "Igor Pro User Files/Igor Procedures". You can locate this folder by choosing Help→Show Igor Pro User Files.

Igor opens global procedure files with write-protection on since they presumably contain procedures that you have already debugged and which you don't want to inadvertently modify. If you *do* want to modify a global procedure file, click the write-protect icon (pencil in lower-left corner of the window). You can turn this default write-protection off via the Text Encoding section of the Miscellaneous Settings dialog.

When you create a new experiment or open an existing one, Igor normally closes any open procedure files, but it leaves global procedure files open. You can explicitly close a global procedure window at any time and then you can manually reopen it. Igor will not automatically reopen it until the next time Igor is launched.

Although its procedures can be used by the current experiment, a global procedure file is not part of the current experiment. Therefore, Igor does not save a global procedure file or a reference to a global procedure file inside an experiment file.

Note: There is a risk in using global procedure files. If you copy an experiment that relies on a global procedure file to another computer and forget to also copy the global procedure file, the experiment will not work on the other computer.

Saving Global Procedure Files

If you modify a global procedure file, Igor will save it when you save the current experiment even though the global procedure file is not part of the current experiment. However, you might want to save the procedure file without saving the experiment. For this, use the File→Save Procedure menu item.

Shared Procedure Files

You may develop procedures that you want to use in several but not all of your experiments. You can facilitate this by creating a shared procedure file. This is a procedure file that you save in its own file, separate from any experiment. Such a file can be opened from any experiment.

There are two ways to open a shared procedure file from an experiment:

- By explicitly opening it, using the File→Open File submenu or by double-clicking it or by drag-and-drop
- By adding an include statement to your experiment procedure window

The include method is preferred and is described in detail under **Including a Procedure File** on page III-354.

When Igor encounters an include statement, it searches for the included file in "Igor Pro 7 Folder/User Procedures" and in "Igor Pro User Files/User Procedures" (see **Igor Pro User Files** on page II-31 for details). The Igor Pro User Files folder is the recommended place for storing user files. You can locate it by choosing Help→Show Igor Pro User Files.

You can store your shared procedure procedure file directly in "Igor Pro User Files/User Procedures" or you can store it elsewhere and put an alias (*Macintosh*) or shortcut (*Windows*) for it in "Igor Pro User Files/User Procedures". If you have many shared procedure files you can put them all in your own folder and put an alias/shortcut for the folder in "Igor Pro User Files/User Procedures".

When you explicitly open a procedure file using the Open File submenu, by double-clicking it, or by drag-and-drop, you are adding it to the current experiment. When you save the experiment, Igor saves a *reference* to the procedure file in the experiment file. When you close the experiment, Igor closes the procedure file. When you later reopen the experiment, Igor reopens the procedure file.

When you use an include statement, the included file is not considered part of the experiment but is still referenced by the experiment. Igor automatically opens the included file when it hits the include statement during procedure compilation.

Note: There is a risk in sharing procedure files among experiments. If you copy the experiment to another computer and forget to also copy the shared files, the experiment will not work on the other computer. See **References to Files and Folders** on page II-22 for more explanation.

Saving Shared Procedure Files

If you modify a shared procedure file, Igor saves it when you save the experiment that is sharing it. However, you might want to save the procedure file without saving the experiment. For this, choose File→Save Procedure.

Including a Procedure File

You can put an include statement in any procedure file. An include statement automatically opens another procedure file. This is the recommended way of accessing files that contain utility routines which you may want to use in several experiments. Using an include statement is preferable to opening a procedure file explicitly because it doesn't rely on the exact location of the file in the file system hierarchy.

Here is a typical include statement:

```
#include <MatrixToXYZ>
```

This automatically opens the MatrixToXYZ.ipf file supplied by WaveMetrics in "Igor Pro 7 Folder/WaveMetrics Procedures/Data Manipulation". The angle brackets tell Igor to search the "Igor Pro 7 Folder/WaveMetrics Procedures" hierarchy.

To see what WaveMetrics procedure files are available, choose Help→Help Windows→WM Procedures Index.

You can include your own utility procedure files by using double-quotes instead of the angle-brackets shown above:

```
#include "Your Procedure File"
```

The double-quotes tell Igor to search the "Igor Pro 7 Folder/User Procedures" and "Igor Pro User Files/User Procedures" hierarchies (see **Igor Pro User Files** on page II-31 for details) for the named file. Igor searches those folders and subfolders and files or folders referenced by aliases/shortcuts in those folders.

These are the two main variations on the include statement. For details on less frequently used variations, see **The Include Statement** on page IV-155.

Included procedure files are not considered part of the experiment but are automatically opened by Igor when it compiles the experiment's procedures.

To prevent accidental alteration of an included procedure file, Igor opens it either write-protected (User Procedures) or read-only (WaveMetrics Procedures). See **Write-Protect Icon** on page III-351.

A #include statement must omit the file's ".ipf" extension:

```
#include <Strings as Lists>      // RIGHT

#include <Strings as Lists.ipf>  // WRONG
```

Creating Packages

A package is a set of procedure files, help files and other support files that add significant functionality to Igor.

Igor comes pre-configured with numerous WaveMetrics packages accessed through the Data→Packages, Analysis→Packages, Misc→Packages, Windows→New→Packages and Graph→Packages submenus as well as others.

Intermediate to advanced programmers can create their own packages. See **Packages** on page IV-232 for details.

Invisible Procedure Files

If you create a package of Igor procedures to be used by regular Igor users (as opposed to programmers), you may want to hide the procedures to reduce clutter or to eliminate the possibility that they might inadvertently change them. You can do this by making the procedure files invisible.

Invisible procedure files are omitted from Igor's Procedure Windows submenu which appears in the Windows menu. This keeps them out of the way of regular users.

There are three ways to make a procedure file invisible. In order of difficulty they are:

- Using the #pragma hide compiler directive
- Using an independent module
- Using the operating-system-supplied file visibility property

Invisible Procedure Windows Using #pragma hide

You can make a procedure file invisible by inserting this compiler directive in the file:

#pragma hide=1

This prevents the procedure window from being listed in the Windows→Procedures submenu. Procedures windows that include this compiler directive become invisible on the next compile.

You can make these windows visible during development by executing:

```
SetIgorOption IndependentModuleDev=1
```

and return them to invisible by executing:

```
SetIgorOption IndependentModuleDev=0
```

You must force a compile for this to take effect.

Prior to Igor Pro 6.30 this feature worked for #included procedure files only, not for packed and standalone procedure files.

Invisible Procedure Windows Using Independent Modules

You can also make a set of procedure files invisible by making them an independent module. The independent module technique is more difficult to implement but has additional advantages. For details, see **The IndependentModule Pragma** on page IV-51.

Invisible Procedure Files Using The Files Visibility Property

This section discusses making procedure files invisible by setting the operating-system-supplied file "visible" property.

Note: This is an old technique that is no longer recommended. It may not be supported in future versions of Igor.

When Igor opens a procedure file, it asks the operating system if the file is invisible (*Macintosh*) or hidden (*Windows*). We will use the term “invisible” to mean invisible on Macintosh and hidden on Windows.

If the file is invisible, Igor makes the file inaccessible to the user. Igor checks the invisible property only when it opens the file. It does not pay attention to whether the property is changed while the file is open.

You create Igor procedures using normal visible procedure files, typically all in a folder or hierarchy of folders. When it comes time to ship to the end user, you set the files to be invisible. If you set a file to be invisible, you should also make it read-only.

You can use the **SetFileFolderInfo** operation (see page V-721) to set the visibility and read-only properties of a file:

```
SetFileFolderInfo /INV=1 /RO=1 "<path to file>"
```

The file will be invisible in Igor the next time you open it, typically by opening an experiment or using a #include statement.

On Macintosh, the file disappears from the Finder when you execute the command.

On Windows, merely setting the hidden property is not sufficient to actually hide the file. It is actually hidden only if the Hide Files of These Types radio button in the View Options dialog is turned on. You can access this dialog by opening a folder in the Windows desktop and choosing View→Options from the folder’s menu bar. Although the hidden property in Windows does not guarantee that the file will be hidden in the Windows desktop, it does guarantee that it will be hidden from within Igor.

After the files are set to be invisible and read-only, if you want to edit them in Igor, you must close them (typically by closing the open experiment), set the files to be visible and read/write again, and then open them again.

Igor’s behavior is changed in the following ways for a procedure file set to invisible:

1. The window will not appear in the Windows→Procedure Windows menu.
2. Procedures in the window will not appear in the contextual pop-up menu in other procedure windows (Control-click on *Macintosh*, right-click on *Windows*).

3. If the user presses Option (*Macintosh*) or Alt (*Windows*) while choosing the name of a procedure from a menu, Igor will do nothing rather than its normal behavior of displaying the procedure window.
4. When cycling through procedure windows using Command-Shift-Option-M (*Macintosh*) or Ctrl+Shift+Alt+M (*Windows*), Igor will omit the procedure window.
5. The Button Control dialog, Pop-Up Menu Control dialog, and other control dialogs will not allow you to edit procedures in the invisible file.
6. The Edit Procedure and Debug buttons will not appear in the Macro Execute Error dialog.
7. If an error occurs in a function in the invisible file and the Debug On Error flag (Procedure menu) is on, the debugger will act as if Debug On Error were off.
8. The debugger won't allow you to step into procedures in the invisible file.
9. The **ProcedureText** function (see page V-662) and **DisplayProcedure** operation (see page V-143) will act as if procedures in the invisible file don't exist. The **MacroList** and **FunctionList** functions will, however work as usual.
10. The GetIgorProcedure and SetIgorProcedure XOPSupport routines in the XOP Toolkit will act as if procedures in the invisible file don't exist. The GetIgorProcedureList function will, however work as usual.

Inserting Text Into a Procedure Window

On occasion, you may want to copy text from one procedure file to another. With the procedure window active, choose Edit→Insert File. This displays an Open File dialog in which you can choose a file and then inserts its contents into the procedure window.

Adopting a Procedure File

Adoption is a way for you to copy a procedure file into the current experiment and break the connection to its original file. The reason for doing this is to make the experiment self-contained so that, if you transfer it to another computer or send it to a colleague, all of the files needed to recreate the experiment are stored in the experiment itself.

To adopt a file, open its associated window and choose File→Adopt Window. This item is available only if the active window is a notebook or procedure file that is stored separate from the current experiment and the current experiment has been saved to disk.

If the current experiment is stored in packed form then, when you adopt a file, Igor does a save-as to a temporary file. When you subsequently save the experiment, the contents of the temporary file are stored in the packed experiment file.

If the current experiment is stored in unpacked form then, when you adopt a file, Igor does a save-as to the experiment's home folder. When you subsequently save the experiment, Igor updates the experiment's recreation procedures to open the new file in the home folder instead of the original file. If you adopt a file in an unpacked experiment and then you do not save the experiment, the new file will still exist in the home folder but the experiment's recreation procedures will still refer to the original file. Thus, you should normally save the experiment soon after adopting a file.

Adoption does not cause the original file to be deleted. You can delete it from the desktop if you want.

To "unadopt" a procedure file, choose Save Procedure File As from the File menu.

It is possible to do adopt multiple files at one time. For details see **Adopt All** on page II-24.

Auto-Compiling

If you modify a procedure window and then activate a non-procedure window other than a help window, Igor automatically compiles the procedures.

If you have a lot of procedures and compiling takes a long time, you may want to turn auto-compiling off. You can do this by deselecting the Auto-compile item in the Macros menu. This item appears only when the procedures need to be compiled (you have modified a procedure file or opened a new one). If you uncheck it item, Igor will not auto-compile and compilation will be done only when you click the Compile button or choose Compile from the Macros menu.

Debugging Procedures

Igor includes a symbolic debugger. This is described in **The Debugger** on page IV-198.

Finding Text

To find text in the active window, choose Edit→Find or press Command-F (*Macintosh*) or Ctrl+F (*Windows*). This displays the Find bar. See **Finding Text in the Active Window** on page II-46 for details.

To search multiple windows for text, choose Edit→Find in Multiple Windows. See **Finding Text in Multiple Windows** on page II-47 for details.

On Macintosh, you can search for the next occurrence of a string by selecting the string, pressing Command-E (Use Selection for Find in the Edit menu), and pressing Command-G (Find Same in the Edit menu).

On Windows, you can search for the next occurrence of a string by selecting the string and pressing Ctrl+H (Find Selection in the Edit menu).

After doing a find, you can search for the same text again by pressing Command-G (*Macintosh*) or Ctrl+G (*Windows*) (Find Same in the Edit menu). You can search for the same text but in the reverse direction by pressing Command-Shift-G (*Macintosh*) or Shift+Ctrl+G (*Windows*).

You can search back and forth through a procedure window by repeatedly pressing Command-G and Command-Shift-G (*Macintosh*) or Ctrl+G and Shift-Ctrl+G (*Windows*).

Replacing Text

To replace text in the active window, press Command-R (*Macintosh*) or Ctrl+R (*Windows*). This displays the Find bar in replace mode. See **Find and Replace** on page II-46 for details.

The Search Selected Text Only option is handy for limiting the replacement to a particular procedure.

While replacing text is undoable, the potential for unintended and wide-ranging consequences is such that we recommend saving the file before doing a mass replace so you can revert-to-saved if necessary.

Another method for searching and replacing consists of repeating Command-F or Ctrl-F (Find) followed by Command-V or Ctrl-V (Paste). This has the virtue of allowing you to inspect each occurrence of the target text before replacing it.

Printing Procedure Text

To print the active procedure window, first deselect all text and then choose File→Print Procedure Window.

To print part of the active procedure window, select the text you want to print and choose File→Print Procedure Selection.

Indentation

We use indentation to indicate the structure of a procedure. This is described in **Indentation Conventions** on page IV-24.

To make it easy to use the indentation conventions, Igor maintains indentation when you press Return or Enter in a procedure window. It automatically inserts enough tabs in the new line to have the same indentation as the previous line.

To indent more, as when going into the body of a loop, press Return or Enter and then Tab. To indent less, as when leaving the body of a loop, press Return or Enter and then Delete. When you don't want to change the level of indentation, just press Return.

Included in the Edit menu for Procedure windows, is the Adjust Indentation item, which adjusts indentation of all selected lines of text to match Igor standards. The Edit menu also contains Indent Left and Indent Right commands that add or remove indentation for all selected lines.

Document Settings

The Document Settings dialog controls settings that affect the procedure window as a whole. You can summon it via the Procedure menu.

Igor does not store document settings for plain text files. When you open a procedure file, these settings are determined by preferences. You can capture preferences by choosing Procedure→Capture Procedure Prefs.

Text Format Settings

You can specify the font, text size, style and color using items in the Procedure menu. Since procedure windows are always plain text windows (as opposed to notebooks, which can be formatted text) these text settings are the same for all characters in the window.

Igor does not store text format settings for plain text files. When you open a procedure file, these settings are determined by preferences. You can capture preferences by choosing Procedure→Capture Procedure Prefs.

Syntax Coloring

Procedure windows and the command window colorize comments, literal strings, flow control, and other syntax elements. Colors of various elements can be adjusted in two ways.

The first is from the Miscellaneous Settings dialog. Select the Text Editing category and then select the Editor Behavior tab.

The second is by executing the following **SetIgorOption** colorize commands::

Command	Effect
`SetIgorOption colorize,doColorize=<1 or 0>`	Turn all colorize on or off
`SetIgorOption colorize,OpsColorized=<1 or 0>`	Turn operation keyword colorization on or off
`SetIgorOption colorize,BIFuncsColorized=<1 or 0>`	Turn function keyword colorization on or off
`SetIgorOption colorize,keywordColor=(`*r*`,`*g*`,`*b*`[,`*a*`])`	Color for language keywords
`SetIgorOption colorize,commentColor=(`*r*`,`*g*`,`*b*`[,`*a*`])`	Color for comments
`SetIgorOption colorize,stringColor=(`*r*`,`*g*`,`*b*`[,`*a*`])`	Color for strings

Command	Effect
`SetIgorOption colorize,operationColor=(r,g,b[,a])`	Color for operation keywords
`SetIgorOption colorize,functionColor=(r,g,b[,a])`	Color for built-in function keywords
`SetIgorOption colorize,poundColor=(r,g,b[,a])`	Color for #keywords such as #pragma
`SetIgorOption colorize,UserFuncsColorized=1`	Turn colorizing on for user functions
`SetIgorOption colorize,userFunctionColor=(r,g,b[,a])`	Color for user-defined functions

Values for *r*, *g*, *b*, and the optional alpha range from 0 to 65535. Alpha defaults to 65535 (opaque).

Changes to syntax coloring settings, made via the dialog or via SetIgorOption, are saved to preferences and used for future sessions.

Procedure Window Preferences

The procedure window preferences affect the creation of *new* procedure windows. This includes the creation of auxiliary procedure windows and the initialization of the built-in procedure window that occurs when you create a new experiment.

To set procedure preferences, set the attributes of any procedure window and then choose Procedure→Capture Procedure Prefs.

To determine the current preference settings, you must create a new procedure window and examine its settings.

Preferences are stored in the Igor Preferences file. See Chapter III-18, **Preferences**, for further information on preferences.

Double and Triple-Clicking

Double-clicking a word conventionally selects the entire word. Igor extends this idea a bit. In addition to supporting double-clicking, if you triple-click in a line of text, it selects the entire line. If you drag after triple-clicking, it extends the selection an entire line at a time.

Matching Characters

Igor includes a handy feature to help you check parenthesized expressions. If you double-click a parenthesis, Igor tries to find a matching parenthesis on the same line of text. If it succeeds, it selects all of the text between the parentheses. If it fails, it beeps. Consider the command

```
wave1 = exp(log(x)))
```

If you double-clicked on the first parenthesis, it would select "`log(x)`". If you double-clicked on the last parenthesis, it would beep because there is no matching parenthesis.

If you double-click in-between adjacent parentheses Igor considers this a click on the outside parenthesis.

Igor does matching if you double-click the following characters:

Left and right parentheses	(xxx)
Left and right brackets	[xxx]
Left and right braces	{xxx}
Plain single quotes	'xxx'
Plain double quotes	"xxx"

Code Comments

The Edit menu for procedures contains two items, Commentize and Decommentize, to help you edit and debug your procedure code when you want to comment out large blocks of code, and later, to remove these comments. Commentize inserts comment symbol at the start of each selected line of text. Decommentize deletes any comment symbols found at beginning of each selected line of text.

Procedure File Text Encodings

Igor Pro 7 uses Unicode internally. Older versions of Igor used non-Unicode text encodings such as MacRoman, Windows-1252 and Shift JIS.

Igor Pro 7 must convert from the old text encodings to Unicode when opening old files. It is not always possible to get this conversion right. You may get incorrect characters or receive errors when opening files containing non-ASCII text.

For a discussion of these issues, see **Text Encodings** on page III-409 and **Plain Text File Text Encodings** on page III-417.

Procedure Window Shortcuts

To view text window keyboard navigation shortcuts, see **Text Window Navigation** on page II-45.

Action	**Shortcut** (*Macintosh*)	**Shortcut** (*Windows*)
To show the built-in procedure window	Press Command-M.	Press Ctrl+M.
To hide the active procedure file and cycle to the next	Press Command-Shift-Option-M.	Press Ctrl+Shift+Alt+M.
To cycle through the open procedure windows	Press Command-Option-M.	Press Ctrl+Alt+M.
To get a contextual menu of commonly-used actions	Press Control and click in the body of the procedure window.	Right-click the body of the procedure window.
To execute commands in a procedure window	Select the commands or click in the line containing the commands and press Control-Return or Control-Enter.	Select the commands or click in the line containing the commands and press Ctrl+Enter.
To insert a template	Type or select the name of an operation, Control-click, and choose Insert Template.	Type or select the name of an operation, right-click, and choose Insert Template.
To get help for a built-in or external operation or function	Type or select the name of an operation, Control-click, and choose Help for <name>.	Type or select the name of an operation, right-click, and choose Help for <name>.
To view the definition of a user-defined function	Type or select the name of an operation, Control-click, and choose Go to <name>.	Type or select the name of an operation, right-click, and choose Go to <name>.
To find a procedure in the active procedure window	Click the navigation bar at the top of the procedure window.	Click the navigation bar at the top of the procedure window.
To find the definition of a procedure when you have selected an invocation of it	Control-click the selected invocation and choose "Go to <procedure name>" from the resulting contextual menu.	Right-click the selected invocation and choose "Go to <procedure name>" from the resulting contextual menu.
To find in the procedure window	Press Command-F.	Press Ctrl+F.
To find the same text again	Press Command-G.	Press Ctrl+G.
To find again but in the reverse direction	Press Command-Shift-G.	Press Ctrl+Shift+G.
To find the selected text	Press Command-E and Command-G.	Press Ctrl+H.
To find the selected text but in the reverse direction	Press Command-E and Command-Shift-G. This shortcut can be changed through the Miscellaneous Settings dialog.	Press Ctrl+Shift+H.

Action	**Shortcut** (*Macintosh*)	**Shortcut** (*Windows*)
To find a user-defined menu's procedure	Open any procedure window and press Option while selecting a user-defined menu item.	Open any procedure window and press Alt while selecting the user-defined menu item.
To select a word	Double-click.	Double-click.
To select an entire line	Triple-click.	Triple-click.