

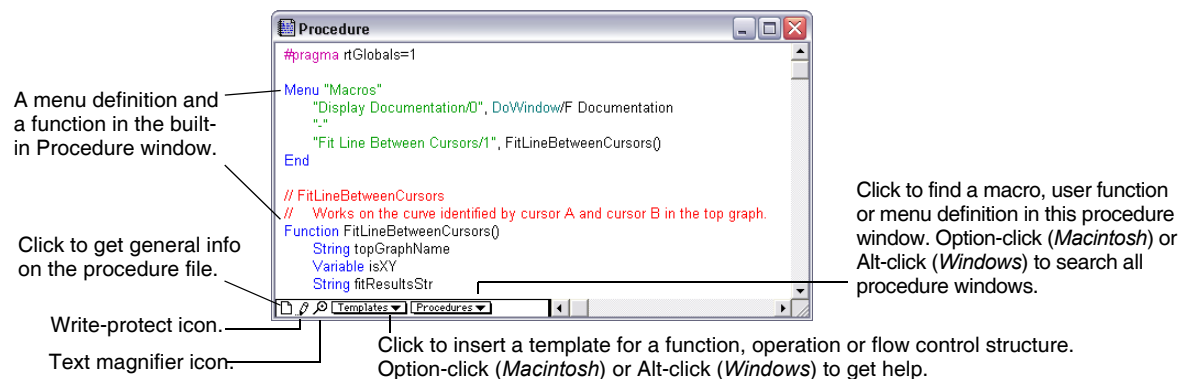
Procedure Windows

- Overview 340
- Types of Procedure Files 340
- Working with the Built-in Procedure Window 340
 - Compiling the Procedures 341
 - Templates Pop-Up Menu 341
 - Procedures Pop-Up Menu 341
 - Magnifier Icon 342
 - Write-Protect Icon 342
- Creating Procedures 342
- Creating New Procedure Files 343
- Opening an Auxiliary Procedure File 343
- Showing Procedure Windows 343
- Hiding and Killing Procedure Windows 344
- Shared Procedure Files 345
 - Saving Shared Procedure Files 345
- Global Procedure Files 345
 - Saving Global Procedure Files 346
- Including a Procedure File 346
- Creating Packages 347
- Invisible Procedure Files 347
 - Invisible Procedure Windows Using #pragma hide 347
 - Invisible Procedure Windows Using Independent Modules 347
 - Invisible Procedure Files Using The Files Visibility Property 347
- Inserting Text 348
- Adopting a Procedure File 349
- Auto-Compiling 349
- Debugging Procedures 349
- Finding Text 349
- Replacing Text 350
- Printing Procedure Text 350
- Indentation 351
- Document Settings 351
- Syntax Coloring 352
- Text Character Settings 352
- Procedure Window Preferences 352
- Double and Triple-Clicking 353
- Matching Characters 353
- Code Comments 353
- UTF-16 Files 353
- Procedure Window Shortcuts 354

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
- Auxiliary experiment procedure files, displayed in auxiliary procedure windows
- Shared procedure files, displayed in auxiliary procedure windows
- Global 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.

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.

A shared procedure file contains procedures that you want to use in more than one experiment but that you don't need in all experiments. It is always saved in a standalone file.

A global procedure file contains procedures that you might want to use in any experiment. It is always saved in a standalone file.

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 Procedure Window from the Procedure Windows submenu of the Windows menu 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*).

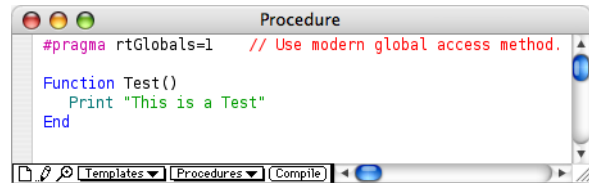
To create a procedure, just type it into the procedure window.

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 they 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.

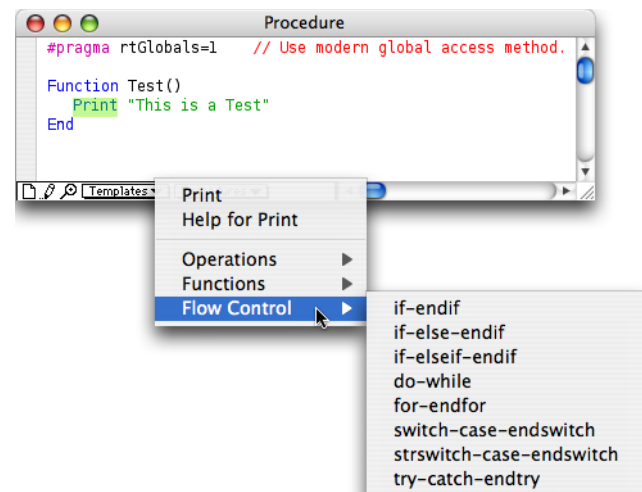
Templates Pop-Up Menu

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.

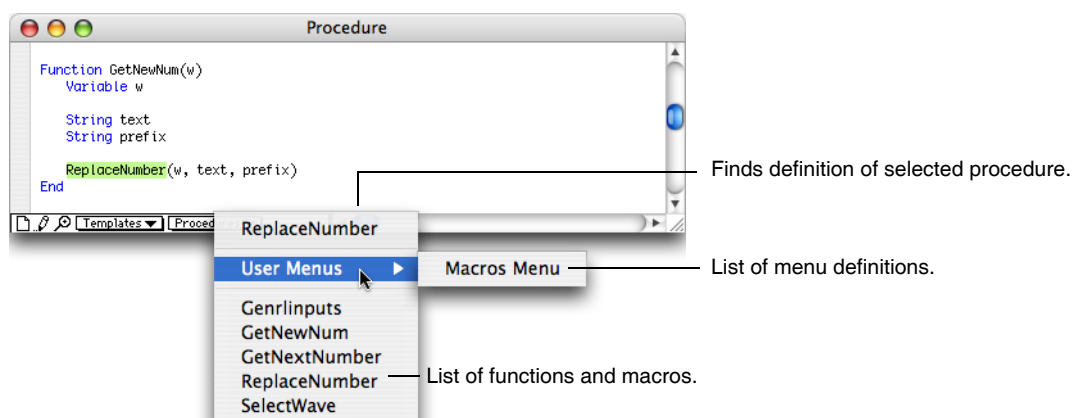
If you select, click in, or click after a recognized operation, function or flow-control keyword in the procedure window, two additional items are listed at the top of the menu. The first item inserts a template and the second takes you to help.

If you select, click in, or click after a term for which Igor can provide a template, an item that inserts a template is added at the top of the menu. If Igor can provide help for the term, another item is added leading to the help. See **Procedure Window Shortcuts** on page III-354.



Procedures Pop-Up Menu

The Procedures pop-up menu provides a quick way for you to find procedures in the procedure window.



If you choose an item, Igor finds it. The menu normally lists just those procedures in the active window but if you press Option (*Macintosh*) or Alt (*Windows*), Igor will include procedures from all open procedure windows.

If you select, click in, or click after a recognized user procedure, an additional item is listed at the top of the menu which finds the selected procedure. Certain kinds of syntax errors prevent Igor from recognizing the selected procedure in which case, the menu will not include an item to find the procedure.

Magnifier Icon

You can magnify procedure text to make it more readable. See **Text Magnification** on page II-71 for details.

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 will draw a line through the pencil, 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.

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 Pro opens WaveMetrics procedures for reading only.

Creating Procedures

There are three ways to create procedures:

- Automatically by Igor
- Manually, when you type in a procedure window
- Semiautomatically, when you use various dialogs (e.g., the dialog that adds controls to a panel and the Curve Fitting dialog)

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. Igor always stores window recreation macros in the built-in procedure window. See **Saving a Window as a Recreation Macro** on page II-61 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.

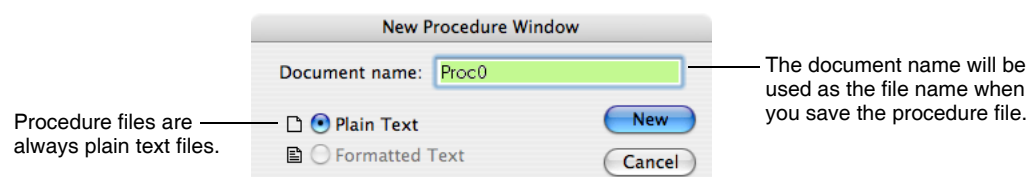
Creating New Procedure Files

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-37 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 Procedure from the New submenu of the Windows menu.



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 will save 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 commonly used auxiliary files, it is better to use the include statement than to explicitly open the files. See **Including a Procedure File** on page III-346.

Showing Procedure Windows

We usually hide procedure windows when we are not doing programming. To show the built-in procedure window, choose Procedure Window from the Procedure Windows submenu of the Windows menu or press Command-M (*Macintosh*) or Ctrl+M (*Windows*). To show auxiliary procedure windows, use the Windows→Procedure Windows menu item.

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.

Chapter III-13 — Procedure Windows

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.

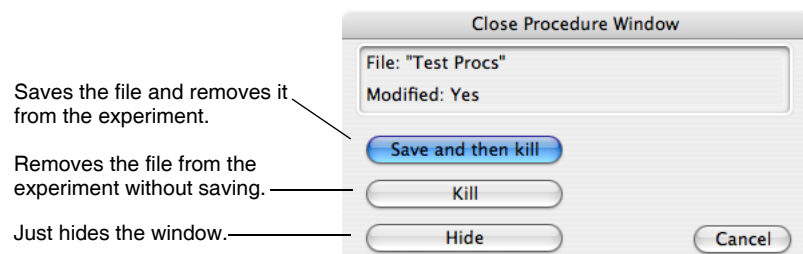
You can show all procedure windows and hide all procedure windows using the Windows→Show and Windows→Hide submenus.

Hiding and Killing Procedure Windows

The built-in procedure window always exists as part of the current experiment. You can hide it by clicking in 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 hide 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, as indicated in these tables.

Macintosh:

Actions	Modifier Key	Result
Click close button, choose Close, press Command-W		Displays dialog
Click close button, choose Close, press Command-W	Shift	Hides window
Click close button, choose Close, press Command-W	Option	Displays dialog

Windows:

Actions	Modifier Key	Result
Click close button, choose Close, press Ctrl+W		Displays dialog
Click close button, choose Close, press Ctrl+W	Shift	Hides window
Click close button, choose Close, press Ctrl+W	Alt	Displays dialog

Macintosh: When the Close Procedure Window dialog is showing, you can press Option to make the Kill button the default. The Kill button will become bold while the “Save and then kill” button will become normal. You can then press Return or Enter to kill the window. Similarly, press Shift to make the Hide button the default button.

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 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-346.

When Igor encounters an include statement, it searches for the included file in "Igor Pro Folder/User Procedures" and in "Igor Pro User Files/User Procedures" (see **Igor Pro User Files** on page II-46 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 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, 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-37 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.

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.

When Igor starts running, it searches "Igor Pro Folder/Igor Procedures" and "Igor Pro User Files/Igor Procedures" (see **Igor Pro User Files** on page II-46 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).

Chapter III-13 — Procedure Windows

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.

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 Folder/WaveMetrics Procedures/Data Manipulation". The angle brackets tell Igor to search the "Igor Pro 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 Folder/User Procedures" and "Igor Pro User Files/User Procedures" hierarchies (see **Igor Pro User Files** on page II-46 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-145.

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-342.

A #include statement must omit the file's ".ipf" extension, if it has one:

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

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

See **Cross-Platform Procedure Compatibility** on page III-404 for details.

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-222 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

In Igor Pro 6.1 or later, 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 but only if the window is opened for read-only or is write-protected using Igor's write-protect icon (see **Write-Protect Icon** on page III-342). Since Igor automatically turns write-protect on for #included files, this method is easy to use with them.

Procedures windows that include this compiler directive and which are read-only or write-protected 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.

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-43.

Invisible Procedure Files Using The Files Visibility Property

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

Chapter III-13 — Procedure Windows

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-557) 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.

The file will still be visible in the Macintosh Finder until you restart the Finder.

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 Procedure pop-up menu at the bottom of all procedure windows.
3. 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*).
4. 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.
5. When cycling through procedure windows using Command-Shift-Option-M (*Macintosh*) or Ctrl+Shift+Alt+M (*Windows*), Igor will omit the procedure window.
6. The Button Control dialog, Pop-Up Menu Control dialog, and other control dialogs will not allow you to edit procedures in the invisible file.
7. The Edit Procedure and Debug buttons will not appear in the Macro Execute Error dialog.
8. 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.
9. The debugger won’t allow you to step into procedures in the invisible file.
10. The **ProcedureText** function (see page V-506) and **DisplayProcedure** operation (see page V-118) will act as if procedures in the invisible file don’t exist. (The **MacroList** and **FunctionList** functions will, however work as usual.)
11. 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

On occasion, you may want to copy text from one procedure file to another. The Insert File item in the Edit menu makes this easy. With the procedure window active, choose Insert File. This will display a dialog in which you can find the file and will then insert 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 will be stored in the experiment itself.

To adopt a file, choose Adopt Window from the File menu. This item will be 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-38.

Auto-Compiling

If you modify a procedure window and then activate a nonprocedure 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.

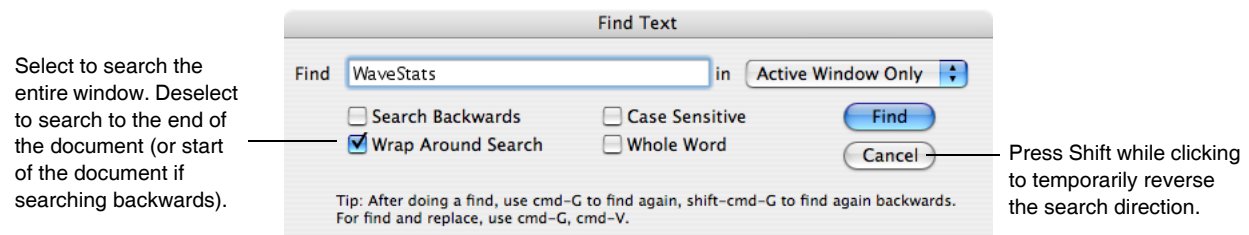
You can do this by deselecting the Auto-compile item in the Macros menu. This item appears only when the procedures need to be compiled (you have modified a procedure file or opened a new one). If you deselect this item, Igor will not auto-compile and compilation will be done only when you click the 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-184.

Finding Text

You can access the Find Text dialog via the Edit menu or by pressing Command-F (*Macintosh*) or Ctrl+F (*Windows*).



Chapter III-13 — Procedure Windows

You can search for the next occurrence of a string by selecting the string and pressing Command-Control-H (*Macintosh*) or Ctrl+H (*Windows*) (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*).

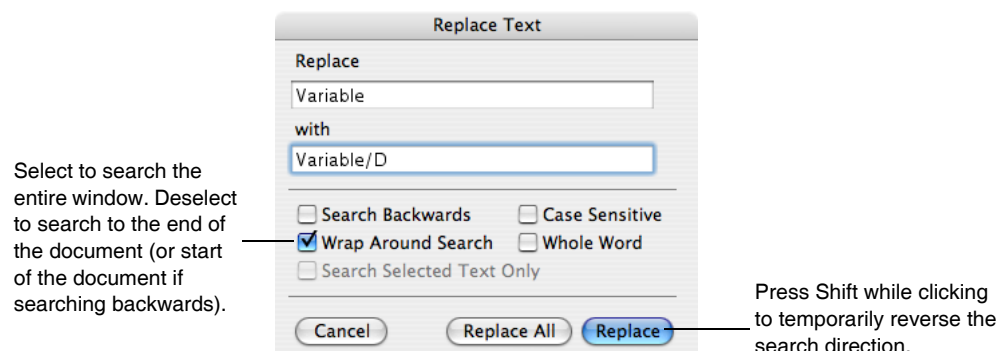
These keyboard shortcuts can be handy. For example, imagine that you are looking at the definition of a function and you want to see where it is used. You can double-click the name of the function to select it and then press Command-Control-H (*Macintosh*) or Ctrl+H (*Windows*) to search for the next occurrence. Then, you can press Command-G (*Macintosh*) or Ctrl+G (*Windows*) to find it again or Command-Shift-G (*Macintosh*) or Ctrl+Shift+G (*Windows*) to go back to the previous occurrence.

You can also perform a Find on multiple help, procedure and notebook windows at one time. See **Finding Text in Multiple Windows** on page II-69.

The Procedures pop-up menu at the bottom of the procedure window provides other ways to find the definition of a procedure in the same or in another procedure file. If you click the pop-up menu, you get a list of all procedures in the active procedure window. If you Option-click (*Macintosh*) or Alt click (*Windows*) on the pop-up menu, you get a list of all procedures in all procedure windows. If you select the name of a procedure and then click the pop-up menu, the first item in the pop-up menu will take you directly to the definition of that procedure, no matter what procedure file it is in.

Replacing Text

You can access the Replace Text dialog via the Edit menu or by pressing Command-R (*Macintosh*) or Ctrl+R (*Windows*).



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

Replacing text is not undoable. Save the file before doing a mass replace so you can revert-to-saved if necessary.

Another method for searching and replacing consists of (*Macintosh*) using Command-F (Find) followed by a series of Command-V (Paste) and Command-G (Find Same) or on *Windows*, Ctrl+F followed by a series of Ctrl+V and Ctrl+G.

Printing Procedure Text

Each procedure file has its own page setup record which you can set using the Page Setup item of the File menu. You can use preferences (see **Procedure Window Preferences** on page III-352) to set the page setup record for new procedure windows.

As of Igor Pro 6.20, to make Igor start up faster, each opened help file and procedure file uses a copy of the same page setup record copied from the preferences for plain text notebooks. This change was made to cope with recent Macintosh HP drivers that take a very long time to create a new page setup record. This means that page setup records stored in help and procedure files loaded during launch are ignored.

You can print all or part of a procedure file.

Macintosh: To print all of the file, click so that no text is selected and then use the Print Procedure File item in the File menu. To print part of it, select the text to be printed and then use the Print Procedure Selection item in the File menu.

Windows: Choose File→Print Procedure File and use the Selection radio button in the Print dialog.

Indentation

We use indentation to indicate the structure of a procedure. This is described in **Indentation Conventions** on page IV-22.

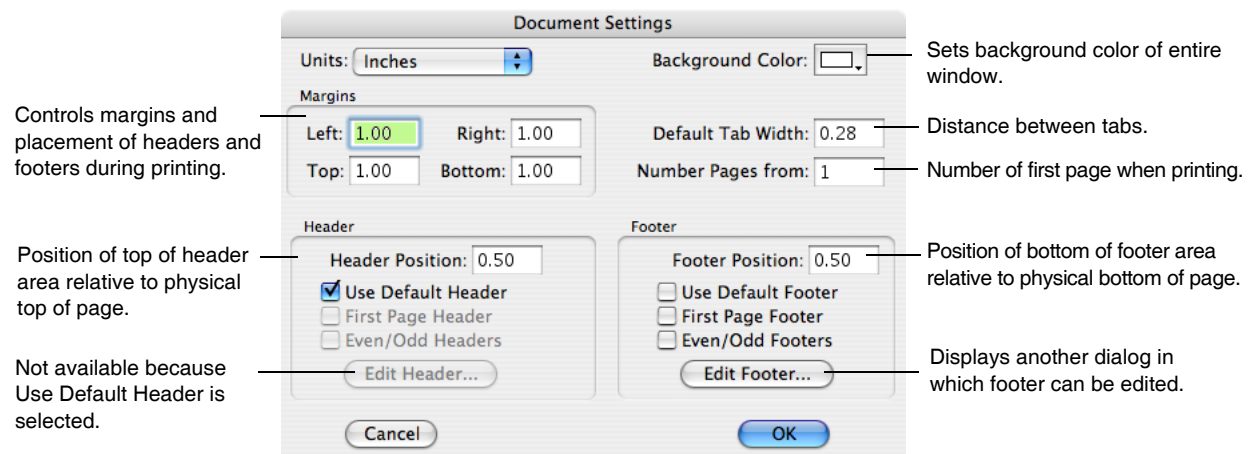
To make it easy to use the indentation conventions, Igor maintains indentation when you press Return or Enter in a procedure window. It automatically inserts enough tabs in the new line to have the same indentation as the previous line.

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.



Note: On *Windows* there is no way for Igor to store settings for an auxiliary procedure file, which is stored as a plain text file. When you open a procedure file or an experiment containing a procedure file on *Windows*, Igor uses preferences to set the procedure file's text format (text font, size, style). Thus, text format changes that you make to a procedure file are lost on *Windows* unless you capture them as your preferred format. The settings for the main procedure window are stored in the experiment file.

Syntax Coloring

The procedure editor colorizes comments, literal strings, flow control, and other function syntax elements. Colors of various elements can be adjusted using the following commands:

Command	Effect
<code>SetIgorOption colorize,doColorize=<1 or 0></code>	Turn all colorize on or off
<code>SetIgorOption colorize,OpsColorized=<1 or 0></code>	Turn operation keyword colorization on or off
<code>SetIgorOption colorize,BIFuncsColorized=<1 or 0></code>	Turn function keyword colorization on or off
<code>SetIgorOption colorize,keywordColor=(<i>r, g, b</i>)</code>	Set color for language keywords
<code>SetIgorOption colorize,commentColor=(<i>r, g, b</i>)</code>	Set color for comments
<code>SetIgorOption colorize,stringColor=(<i>r, g, b</i>)</code>	Set color for strings
<code>SetIgorOption colorize,operationColor=(<i>r, g, b</i>)</code>	Set color for operation keywords
<code>SetIgorOption colorize,functionColor=(<i>r, g, b</i>)</code>	Set color for built-in function keywords
<code>SetIgorOption colorize,poundColor=(<i>r, g, b</i>)</code>	Set color for #keywords such as #pragma

The settings last only for the length of the Igor session. See **SetIgorOption** operation on page V-562.

Values for *r*, *g*, and *b* range from 0 to 65535.

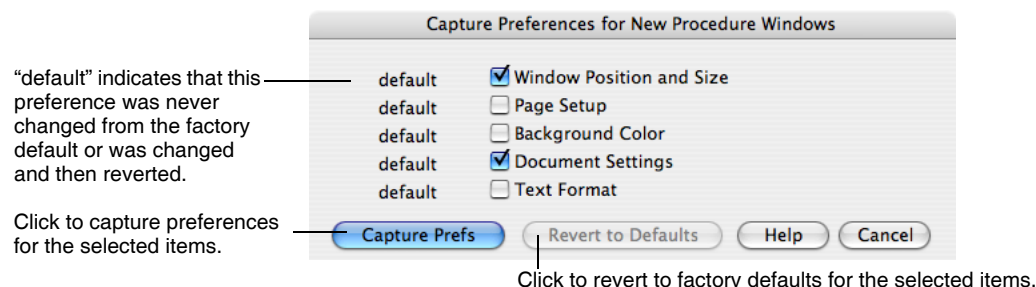
Text Character Settings

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. On Macintosh, text format settings are stored in the resource fork of the procedure file. On Windows, where there is no resource fork, text format settings are not stored and revert to the preferred settings when a procedure window is opened.

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 use the Capture Procedure Prefs item in the Procedure menu.



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

Procedure windows each have their own Page Setup values. New procedure windows will have their own copy of the captured (or default) Page Setup values.

Preferences are stored in the Igor Preferences file. See Chapter III-17, **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"
Typographic single quotes	‘xxx’*
Typographic double quotes	“xxx”*

* No matching is done on two-byte typographic quotes, which are used in Asian fonts.

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.

UTF-16 Files

You can open UTF-16 (two-byte Unicode) text files as procedure windows. Igor does not recognize non-ASCII characters, but does ignore the byte-order mark at the start of the file (BOM) and null bytes contained in UTF-16 text files. If you open a UTF-16 file and then save it from Igor, it will be saved as plain ASCII, not UTF-16, and some information may be lost.

Procedure Window Shortcuts

To view text window keyboard navigation shortcuts, see **Text Window Navigation** on page II-68.

Action	Shortcut (<i>Macintosh</i>)	Shortcut (<i>Windows</i>)
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 or function and press Shift-Help.	Type or select the name of an operation or function and press Ctrl+F1.
	Control-click the name of an operation or function.	Right-click the name of an operation or function.
To get help for an operation or function	Type or select the name of an operation or function and press Shift-Option-Help.	Type or select the name of an operation or function and press Ctrl+Alt+F1.
	Control-click the name of an operation or function.	Right-click the name of an operation or function.
To find a procedure in the active procedure window	Click the Procedures pop-up menu at bottom of the procedure window.	Click the Procedures pop-up menu at bottom of the procedure window.
To find a procedure in any procedure window	Press Option and click the Procedures pop-up menu at bottom of any procedure window.	Press Alt and click the Procedures pop-up menu at bottom of any procedure window.
To find the definition of a procedure when you have selected an invocation of it	Click the Procedures pop-up menu at bottom of the procedure window or press Shift-Option-Help.	Click the Procedures pop-up menu at bottom of the procedure window or press Ctrl+Alt+F1.
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-Control-H.	Press Ctrl+H.
To find the selected text but in the reverse direction	Press Command-Control-Shift-H.	Press Ctrl+Shift+H.

Action	Shortcut (<i>Macintosh</i>)	Shortcut (<i>Windows</i>)
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.
