

Chapter
III-16

Miscellany

- Dashed Lines 410
- The Color Environment 410
- Miscellaneous Settings 411
 - Graph Settings 411
 - Table Settings 411
 - Command Settings 411
 - Experiment Settings 412
 - Data Loading Settings 412
 - Color Settings (Macintosh) 412
 - Typography Settings (Macintosh) 412
 - Asian Language Settings (Macintosh) 413
 - Compatibility Settings 414
 - Misc Settings 414
 - Help Settings (Windows) 415
- Object Names 415
 - Standard Object Names 415
 - Liberal Object Names 415
 - Name Spaces 416
- Renaming Objects 416
- The Object Status Dialog 417
 - User Functions 419
 - Broken Objects 420
- Graphics Technology 421
- Pictures 421
 - Importing Pictures 422
 - The Picture Collection Stores Named Pictures 422
 - Pictures Dialog 423
 - NotebookPictures 423
- Igor Extensions 423
 - WaveMetrics XOPs 423
 - Third Party Extensions 424
 - Activating Extensions 424
 - XOPs on Intel Macintosh 424
 - Running PowerPC XOPs on Intel Macintosh 425
- Memory Management 425
 - Increasing Virtual Memory Space in Windows VISTA and Windows 7 425
 - Increasing Virtual Memory Space in Windows XP 426
- Macintosh System Requirements 426
- Windows System Requirements 426
- Crashes 426
 - Crash Logs on Mac OS X 427
 - Crashes On Windows 427

Dashed Lines

The dashed lines built into Igor can be edited with the Dashed Lines dialog (Misc menu).

The screenshot shows the "Dashed Lines" dialog box. It features a table with columns for "Pair", "Dash", and "Gap". The "Dash" column contains values like 17, 17, 5, 5, 5, and the "Gap" column contains 3, 2, 2, 2, 2. A "Dashed Line" dropdown menu is set to 17, and the "Units" are set to "points". A ruler shows a scale from 0 to 300. Below the table are buttons for "Fewer Pairs", "More Pairs", "Default Dashed Line 17", and "Revert Dashed Line 17". There is also a checkbox for "Capture as Prefs for New Experiments" and a "Set Dashed Lines" button. Annotations with arrows point to various parts of the dialog:

- "Enter the length of a drawn segment." points to the "Dash" input field.
- "Enter the length of the following gap." points to the "Gap" input field.
- "Choose one of 17 dashed lines to edit." points to the "Dashed Line" dropdown menu.
- "Choose the units used in the ruler and for the Dash and Gap values." points to the "Units" dropdown menu.
- "Click to return the current line to the factory default." points to the "Default Dashed Line 17" button.
- "Click to return the current line to what it was before this dialog was brought up." points to the "Revert Dashed Line 17" button.
- "Drag a handle or enter a number to change a length." points to the "Dash" and "Gap" input fields.
- "You can specify up to 8 pairs." points to the "More Pairs" button.
- "If this is selected, new experiments will have the same 17 dashed lines." points to the "Capture as Prefs for New Experiments" checkbox.

You can edit one dashed line pattern at a time. The example shows the last dashed line (number 17) being edited. You can select another line with the Dashed Line pop-up menu. Dashed line 0 (the solid line) cannot be edited. If you need to create a custom dashed line pattern, we recommend that you modify the high numbered dashed lines, leaving the low number ones in their default state. This ensures that the low numbered patterns will be the same for everyone.

Dashed lines are described by pairs of dash and gap lengths. You can add or remove pairs with the More Pairs and Fewer Pairs buttons. The lengths can be adjusted by entering dash and gap values, or by dragging the dash (filled boxes) and gap (hollow boxes) handles.

You can also change dashed lines with the **SetDashPattern** operation (see page V-552).

Dashed lines are stored with the experiment, so each experiment may have different dashed lines. You can capture the current dashed lines as the preferred dashed lines for new experiments.

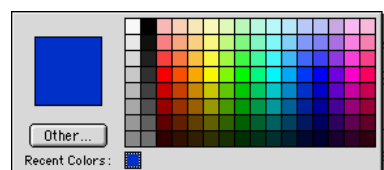
The Color Environment

Igor has a main color palette that contains colors that you can use for traces in graphs, text, axes, backgrounds and so on. The main color palette appears as a pop-up menu in a number of dialogs, such as the Modify Trace Appearance dialog. This section discusses this palette.

Igor also has color tables and color index waves you can select among when displaying contour plots and images. These are discussed in Chapter II-14, **Contour Plots**, and Chapter II-15, **Image Plots**.

You can select from colors presented in a color palette.

You can use the Other button to select colors that are not in the palette. As you use Igor, colors are added to the palette in the Recent Colors area.

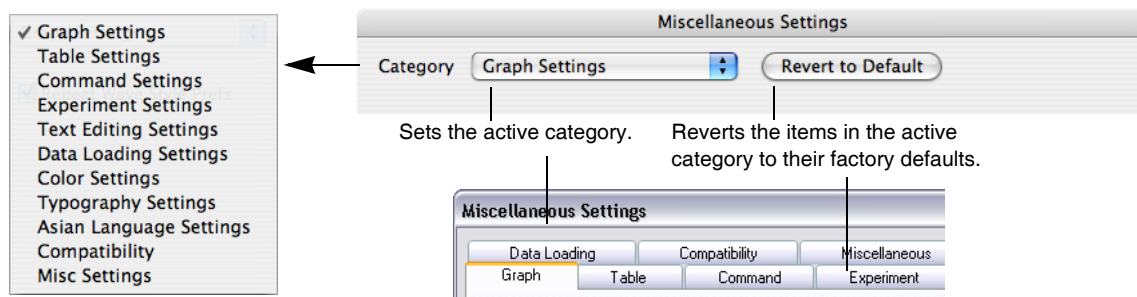


On Macintosh only, the recent colors are remembered by Igor when it quits and restored when it restarts if you have selected the "Save and restore color palette's recent colors" checkbox in the Miscellaneous Settings dialog's Color Settings category.

If you run your monitor in 256 (8 bit) color mode, Igor has to make compromises in an attempt to provide reasonably accurate colors while still allowing other programs to display reasonably accurate colors. It is recommended that you set your monitor to the thousands of colors (16 bit) or millions of colors (32 bit) mode.

Miscellaneous Settings

You can customize some aspects of how Igor works using the Miscellaneous Settings dialog. The miscellaneous settings are grouped into categories.



Changes to miscellaneous settings take effect immediately when you click the Save Settings button. They affect the current experiment as well as new and old (reopened) experiments.

Graph Settings

The Repeat Wave Style Prefs in Graphs checkbox affects the way wave trace style preferences are applied when waves are appended to a graph. The factory default is selected. See **Graph Preferences** on page II-298.

When the Enable “Fling” Mode checkbox is checked, dragging in a graph with Option or Alt pressed and the mouse down will “fling” (scroll) the graph contents in the direction of the drag. To stop the motion, click in the graph while keeping the mouse still.

Table Settings

The Repeat Column Style Prefs in Tables checkbox affects the way column style preferences are applied when columns are appended to a table. The factory default is selected. See **Table Preferences** on page II-228.

Command Settings

The Limit Command History Text checkbox determines the number of lines of history text that Igor will retain in a given experiment. The factory default is deselected (no limit). Limiting history can save space on disk but more importantly can reduce clutter in the history area of the command window. It can also reduce the time it takes to open and save an experiment.

If you select the Limit Command History Text checkbox and enter the number of lines of history that you want to keep, Igor will automatically trim the history to that number of lines when you save the experiment. In addition, when Igor adds text to the history, it checks the number of lines. If it exceeds the requested maximum by 500 lines, Igor trims the history at that time, rather than waiting until you save the experiment. This is done so that if, for example, you run an experiment overnight, the history will not grow without bound.

You can enter and execute commands in a notebook as well as from the command window.

When entering a command via a notebook

- Copy the command to the history area
- Copy command output to the history area

These settings let you determine whether Igor stores commands and output in the history area as well as in the notebook. See **Notebooks as Worksheets** on page III-5 for general information on using a notebook as a worksheet.

Experiment Settings

The Saved Experiment Format pop-up menu controls the way Igor experiments are saved. You can override this choice in the Save Experiment dialog. The choice here merely presets the dialog's Packed checkbox. The factory default is Packed. See **Saving Experiments** on page II-29 for details.

Files and Folder
Packed

Unlike most spreadsheet and graphing programs, data can exist in Igor without being visible in a table. The “Make a new table for new experiments” checkbox controls whether you will get a new empty table when you choose New Experiment from the File menu. The default state is selected. This is convenient if you generally start working by entering data manually. If you do not enter data manually, you may find it more convenient to turn this checkbox off and create a table only when you need one.

Data Loading Settings

The Loaded Igor Binary Data pop-up menu determines how Igor handles the loading of Igor Binary waves from disk files. This menu addresses a subtlety that caught many early Igor users off guard. Igor Binary data can be shared between experiments. Old versions of Igor defaulted to *sharing* loaded Igor Binary data, but many users thought that a loaded wave was another isolated *copy* of the data.

Share (do not copy)
Always Copy to Home
Ask if Copy to Home

Use this pop-up menu to choose how Igor handles Igor Binary waves loaded from a disk file. Copy to Home makes an isolated copy of the wave in the experiment's home folder. If the experiment is saved in packed format, the isolated copy is stored in the experiment's packed file, which contains all the things usually stored in a home folder.

If you choose Ask if Copy to Home, you always get a chance to change your mind with a dialog. If you choose Share or Always Copy to Home, you can change your mind only if you use the Load Waves dialog; the Load Igor Binary dialog uses the setting you make here without asking for confirmation. The factory default is Ask if Copy to Home. See **Sharing Versus Copying Igor Binary Files** on page II-165 for details.

Default Data Precision presets the numerical precision checkboxes in the Make Waves and Load Waves dialogs. The factory default is Double. Note that this affects only the dialogs; if you use commands entered manually on the command line the default is single precision. See **Number Type and Precision** on page II-81 for details.

Single
Double

Color Settings (Macintosh)

Selecting the “Save and restore color palette's recent colors” checkbox restores the recent colors that were in the color palette when Igor quit. If deselected, Igor starts up with no recent colors. The factory default is deselected (don't restore colors).

The Color Palette menu sets the accuracy with which colors are drawn when you run in 256 color (8 bit) mode. It is recommended that you set your monitor to the thousands of colors (16 bit) or millions of colors (32 bit) mode.

Typography Settings (Macintosh)

This section discusses the way Igor draws text in graphs and page layouts.

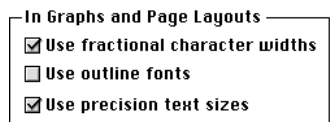
By default, Igor handles text in graphs and page layouts in a method designed to provide good readability of text on the screen and also good fidelity between text on the screen and text on the printed page. If you are satisfied with Igor's default performance in these regards, then you don't need to read the following material.

There is a trade-off to be made between readability of text on the screen and fidelity of text between the screen and the printed page. Unfortunately, there is no way to have maximum readability and maximum fidelity at the same time. You can control this trade-off via three checkboxes in the Typography section of the Miscellaneous Settings dialog.

Achieving text fidelity requires certain actions on the part of Igor, Apple's QuickDraw graphics software and the printer driver. Lack of fidelity stems from the fact that screen resolution is low while printer resolution is high. Because of roundoff and other effects, the width of a string on the screen does not necessarily

match the width of that same string when printed. The reasons for this are too complex to explain here. Apple’s technical note TN-TEXT TE 21 gives some idea of the issues involved.

The “Use fractional character widths”, “Use outline fonts”, and “Use precision text sizes” checkboxes control aspects of how Igor draws text in graphs and page layouts.



The default and recommended state for the “Use fractional character widths” and “Use precision text sizes” checkboxes is on. Turning “Use outline fonts” on makes text in some fonts, such as Geneva and Monaco, difficult to read on screen. Because of this, the default state for this checkbox is off.

When the “Use fractional character widths” checkbox is on, Igor uses Apple’s QuickDraw to keep track of the position of characters using higher precision arithmetic. This slightly distorts text on the screen but improves the fidelity with printed text.

When the “Use outline fonts” checkbox is on, Igor uses Apple’s QuickDraw to create the outline versions of fonts (TrueType or PostScript) instead of the bitmap versions. This also distorts text on the screen but improves the fidelity with printed text. The “Use outline fonts” checkbox makes a big difference with some fonts, such as Geneva, Monaco and Courier, and makes little difference with other fonts, such as Helvetica, Palatino and Times. Results on your computer may be different because you have different versions of these fonts, different system software or different printer driver software.

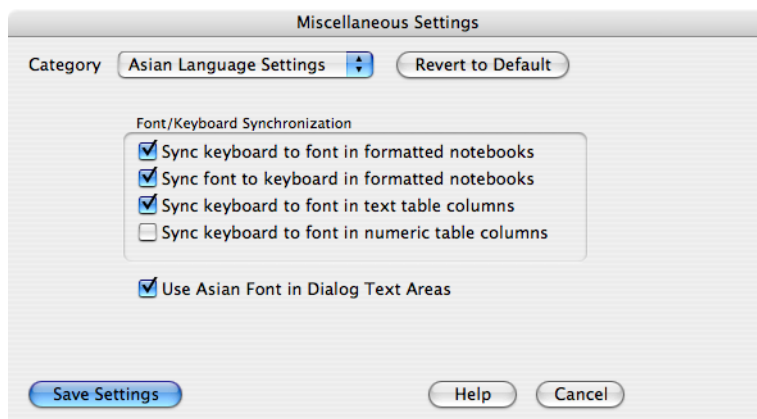
The “Use precision text sizes” checkbox to do a more precise calculation when converting screen text sizes to printer text sizes. The only reason to turn this feature off is to make this version of Igor behave like Igor Pro 2.0.

If all three checkboxes are deselected, you will get the same behavior as in Igor Pro 2.0. This produces good looking text but with less emphasis on text fidelity. Use it only for backward compatibility.

If you change the state of the “Use fractional character widths” or “Use outline fonts”, Igor will automatically update all open graphs and page layouts. This is not needed if you change just the Use precision text sizes checkbox because this does not affect what is drawn on the screen.

Asian Language Settings (*Macintosh*)

The Font/Keyboard Synchronization settings control whether or not Igor will automatically match the keyboard input method and the active font. Synchronization applies to text documents and to tables.



The “Sync keyboard to font in text windows” checkbox determines whether Igor will set the keyboard script when you choose a font or click in a section of text in a particular font. If this is selected and you click in Asian text, Igor will activate the Asian keyboard method. If you click in Roman text, Igor will deactivate the Asian keyboard method. Igor chooses the Asian keyboard method if the font is Asian and the text at the insertion point in the document is Asian. Thus, if you mix Asian and Roman characters all in an Asian font, Igor will activate the appropriate keyboard for the text at the insertion point.

The “Sync font to keyboard in text windows checkbox” determines whether Igor will change the font automatically if you manually change the keyboard input method. If this is selected and you change the keyboard input method to Asian, Igor will automatically switch to an Asian font. If you change the keyboard input method to Roman, Igor will automatically switch to a Roman font. You may prefer to enter Roman text in an Asian font. If so then you should deselect this checkbox and manually change the font as needed.

The “Sync keyboard to font in table text columns” checkbox determines whether Igor will set the keyboard script when you click in a text column in a table. If this is on and you click in a column whose font is Asian, Igor will activate the Asian keyboard methods. If you click in a column whose font is Roman, Igor will deactivate the Asian keyboard method.

The “Sync keyboard to font in table numeric columns” checkbox determines whether Igor will set the keyboard script when you click in a numeric column in a table. We recommend that you leave this deselected and use Roman numbers in numeric columns because Igor currently does not understand non-Roman numbers.

The “Use Asian Font for Dialog Text Areas” checkbox controls the font that Igor will use for certain special dialog text entry areas, such as the text entry area in the Add Annotation and Browse Waves dialogs. Because they require advanced capabilities such as scrolling and undo, these areas are not standard Macintosh text boxes but rather are implemented using WaveMetrics’ special routines. Normally these areas use the Geneva font. This prevents Asian users from entering Asian characters. When the Use Asian Font for Dialog Text Areas checkbox is selected, Igor will use an Asian font for these areas.

Compatibility Settings

The “Native GUI Appearance for User-defined Controls” checkbox sets the default user-defined control appearance. Deselect it to use the Igor Pro 5 user-defined control appearance.

Deselecting the Floating Tool Palettes and Floating Graph Info Palettes checkboxes sets the tool and info palette locations to be internal to the associated window, as in Igor Pro 5, rather than using the default floating palettes. The settings will affect only new ShowInfo and ShowTools commands; current info and tool palette locations are not affected.

Misc Settings

The “Use short menu bar names” checkbox, when selected, shortened some menu names. If deselected, it uses long menu names. The factory default is deselected (long names). You may want to select this if you are running on a small screen or have many user-defined menus.

The “Maximum pop-up menu items” setting limits the number of menu items in some pop-up menus, such as the pop-up menus of waves, variables, string variables, and the Current Object pop-up menu in the Object Status dialog. Choosing a small value (the minimum is 50) prevents excessive start-up time for dialogs at the expense of listing all the items in the pop-up menus. We recommend that you set this to several hundred. Igor displays the spinning hand watch cursor when building large pop-up menus. You can type Command-period (*Macintosh*) or Ctrl+Break (*Windows*) to stop building the pop-up menu. Some dialogs have multiple pop-up menus that cause the cursor to reappear; you can type Command-period (*Macintosh*) or Ctrl+Break (*Windows*) again.

“Preferred units of length” controls the default units used in the Modify Graph, Print Graph, Export Graph, and Save Graphics dialogs. It does not affect page layouts or notebooks. These have their own preference settings for units (use the Capture Layout Preferences and Capture Notebook Preferences dialogs). This also does not affect the control dialogs (for adding buttons, checkboxes, etc.). These dialogs always use points as the default unit of measure.

points inches cm

“Operations that overwrite or delete folders” determines what to do when a MoveFolder or CopyFolder command is about to overwrite a folder on disk or when a DeleteFolder command is about to delete a folder on disk.

The factory default setting is “Display dialog to ask user for permission”. This means that Igor will ask for permission each time one of these operations is about to overwrite or delete a folder on disk. This setting is intended to reduce the chance of accidental or malicious deletion of files.

The other choices are “Always give permission” and “Always deny permission”. “Always give permission” grants blanket permission to overwrite or delete folders. Choosing this increases the risk of accidental or malicious deletion of files so you should exercise caution.

(*Macintosh*) “Use Command-H for Find Selection (instead of Command-Control-H)” controls the keyboard shortcut for the “Find Selection” menu item in the Edit menu. The Find Selection keyboard shortcut defaults to Command-Control-H, allowing Command-H to hide Igor on Mac OS X, per Apple’s recommendation. The Igor Pro 4 keyboard shortcut of Command-H can be restored by selecting this checkbox. When selected, hiding Igor on Mac OS X requires selecting “Hide Igor” with the mouse.

Keyboard Navigation affects the behavior of keys such as Page Down, Page Up, End and Home.

Use Macintosh Conventions Use Windows Conventions

The Windows Menu Shows popup menu determines how target windows are identified in the submenus of the Windows menu. By default, just the window’s title is displayed. You can choose to display the title and/or the name using this setting.

Help Settings (*Windows*)

These are controls governing the display of tool tips, the little windows that pop up automatically when you point at buttons and icons.

Object Names

Every Igor object has a name which you give to the object when you create it or which Igor automatically assigns. You use an object’s name to refer to it in dialogs, from commands and from Igor procedures. The named objects are:

Waves	Data folders	Variables (numeric and string)
Windows	Symbolic paths	Pictures
Annotations	Controls	Rulers

In Igor Pro, the rules for naming waves and data folders are not as strict as the rules for naming all other objects, including string and numeric variables, which are required to have standard names. These sections describe the standard and liberal naming rules.

Standard Object Names

Here are the rules for standard object names:

- May be 1 to 31 characters in length.
- Must start with an alphabetic character (A-Z or a-z).
- May include alphabetic or numeric characters or the underscore character.
- Must not conflict with other names (of operations, functions, etc.).

All names in Igor are case insensitive. wave0 and WAVE0 refer to the same wave.

Characters other than letters and numbers, including spaces and periods, are not allowed. We put this restriction on names so that Igor can identify them unambiguously in commands, including waveform arithmetic expressions.

Historical Note:

Prior to Igor Pro 3.0, wave names were limited to 18 characters.

Liberal Object Names

The rules for liberal names are the same as for standard names except that almost any character can be used in a liberal name. Liberal name rules are allowed for waves and data folders only.

Chapter III-16 — Miscellany

If you are willing to expend extra effort when you use liberal names in commands and waveform arithmetic expressions, you can use wave and data folder names containing almost any character. If you create liberal names then you will need to enclose the names in single (not curly) quotation marks whenever they are used in commands or waveform arithmetic expressions. This is necessary to identify where the name ends. Liberal names have the same rules as standard names except you may use any character except control characters and the following:

" ' : ;

Here is an example of the creation and use of liberal names:

```
Make 'wave 0';    // 'wave 0' is a liberal name
'wave 0' = p
Display 'wave 0'
```

Note: Providing for liberal names requires extra effort and testing on the part of Igor programmers (see **Programming with Liberal Names** on page IV-147) so you may occasionally experience problems using liberal names with user-defined procedures.

Name Spaces

When you refer to an object by name, in a user function for example, each object must be referenced unambiguously. In general, an object must have a unique name so. Sometimes the object type can be inferred from the context, in which case the name can be the same as objects of other types. Objects whose names can be the same are said to be in different name spaces.

Data folders are in their own name space. Therefore the name of a data folder can be the same as the name of any other object, except for another data folder at the same level of the hierarchy.

Waves and variables (numeric and string) are in the same name space and so Igor will not let you create a wave and a variable in a single data folder with the same name.

An annotation is local to the window containing it. Its name must be unique only among annotations in the same window. The same applies for controls and rulers. Data folders, waves, variables, windows, symbolic paths and pictures are global objects, not associated with a particular window.

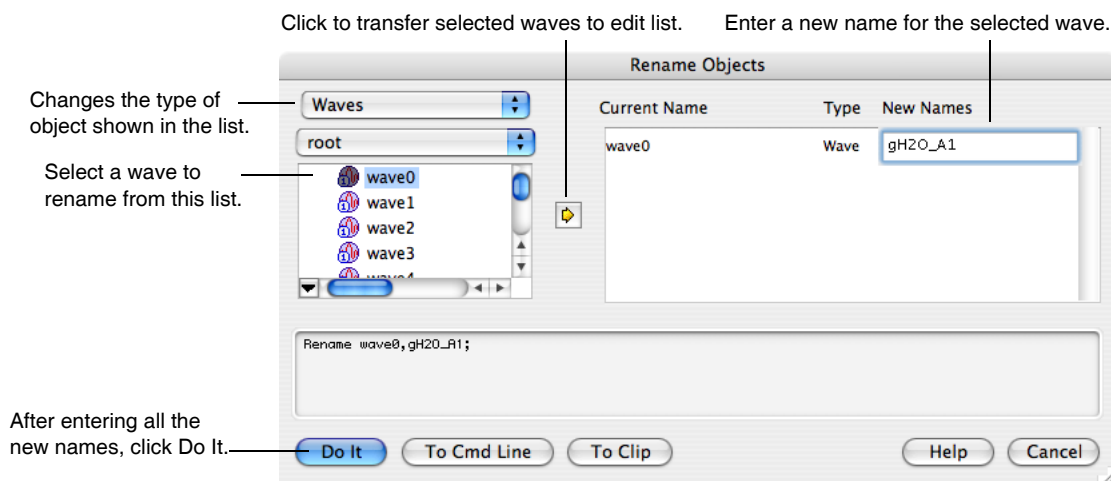
The names of global objects, except for data folders, are required to be distinct from the names of macros, functions (built-in, external or user-defined) and operations (built-in or external).

Here is a summary of the four global name spaces:

Name Space	Requirements
Data folders	Names must be distinct from other data folders at the same level of the hierarchy.
Waves, variables, windows	Names must be distinct from other waves, variables (numeric and string), windows.
Pictures	Names must be distinct from other pictures.
Symbolic paths	Names must be distinct from other symbolic paths.

Renaming Objects

You can use Misc→Rename Objects or Data→Rename to rename waves, variables, strings, symbolic paths, and pictures. Both of these invoke the Rename Objects dialog.



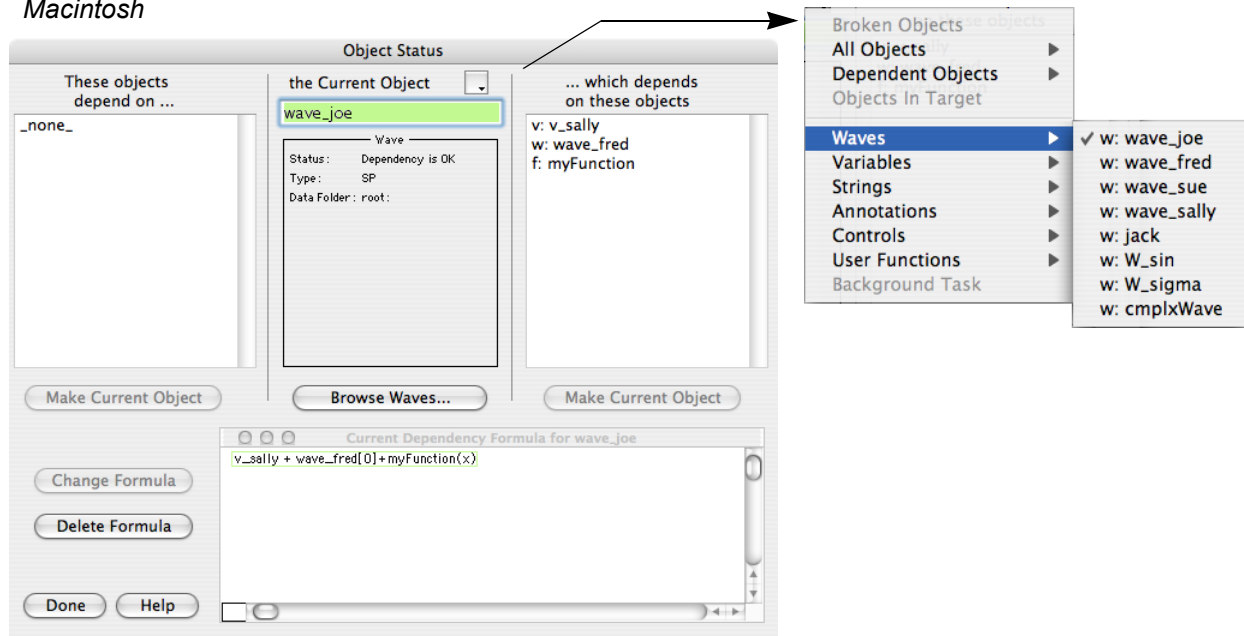
Graphs, tables, page layouts, notebooks, control panels and XOP target windows (e.g., Gizmo 3D plots) are renamed using the **DoWindow** operation (see page V-122) which you can build using the Window Control dialog. See **The Window Control Dialog** on page II-64.

You can use the DataBrowser to rename data folders, waves, and variables. See **Data Browser** on page II-130.

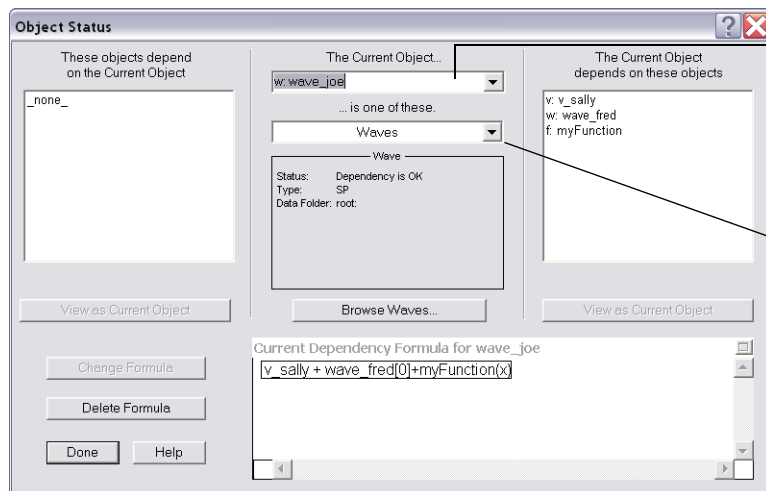
The Object Status Dialog

You can examine the status and interdependencies of named Igor objects with the Object Status dialog. See Chapter IV-9, **Dependencies**, for a discussion of object dependencies. You won't need to use this dialog unless you've got a fairly complex Igor experiment with lots of objects.

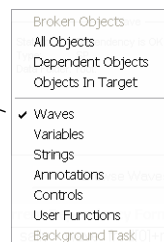
Macintosh



Windows



Choose an object from the Current Object drop-down list, which displays objects of a type chosen in this pop-up menu.



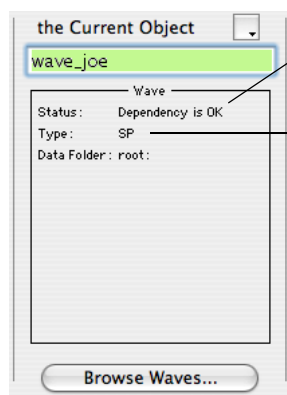
The dialog displays the properties of one object, called the “Current Object”. The name of the current object appears at the top center of the dialog. You can choose a new current object from the Current Object pop-up menu.

You can also choose a new current object by selecting an object from either of the two **dependency lists** on the left and right of the current object and clicking View as Current Object, or by double-clicking in either list.

Items in the lists and submenus are object names preceded by a key indicating the type of object:

Key	Object Type	Key	Object Type
a:	annotation	t:	task (background task)
c:	control	v:	variable
f:	function	w:	wave
s:	string	=:	dependency formula

The current value, dependency status, and other information about the current object is displayed in the box below the name of the current object. Depending on the type of object, a button may also be present for editing or examining that object.



Dependency status, if any

SP means “Single Precision Floating Point”.

DP means “Double Precision Floating Point”.

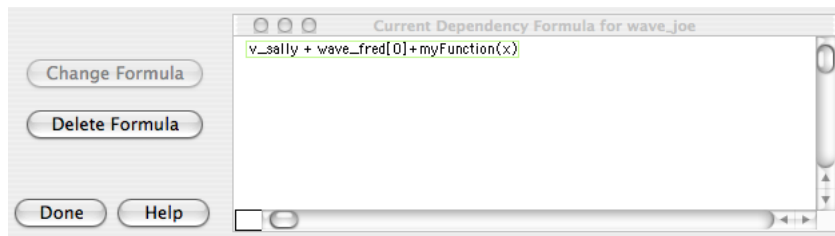
INT32 means “Integer, 32 bits” (also INT8 and INT16).

UINT32 means “Unsigned Integer, 32 bits” (also UINT8 and UINT16).

CMLPX means “Complex” (has real and imaginary values).

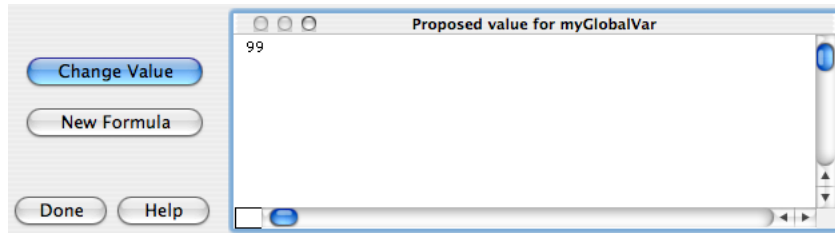
TEXT means text.

Other current object information is displayed in a windoid near the bottom of the dialog. If the object is a “dependent” object, the dependency formula is shown. This formula can be changed or deleted by clicking the appropriate buttons:

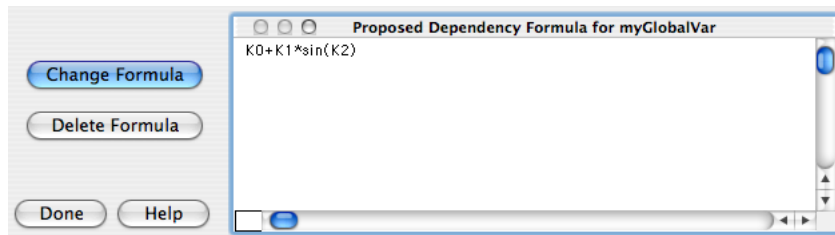


Dependencies and dependency formulas are explained further in Chapter IV-9, **Dependencies**.

You can edit the current value of most nondependent objects in the windoid:



For some nondependent objects, you can also establish a new dependency formula. Click the New Formula button, and an empty formula is created. Enter the desired formula and click Change Formula:



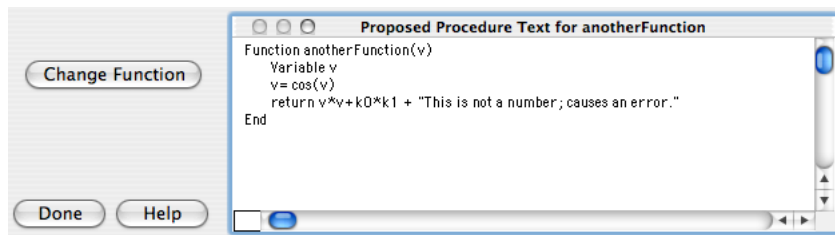
This example established the dependency formula:

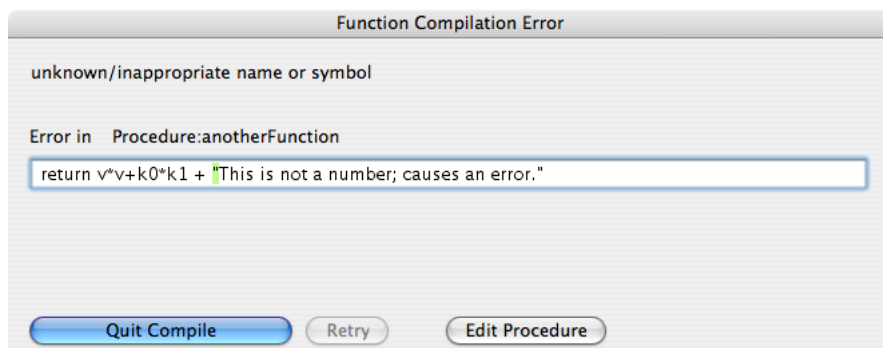
```
myGlobalVar := K0+K1*sin(K2)
```

Whenever K0, K1, or K2 changes, myGlobalVar will be updated using this formula.

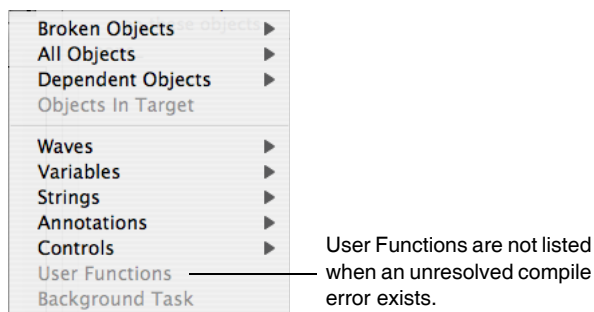
User Functions

You can edit the “current value” of a user-defined function, too. However, if you introduce an error and click Change Function, the function compiler will display an error dialog.





If you then click Quit Compile, the Object Status dialog will come back up, but it will appear as if all the user-defined functions have vanished.

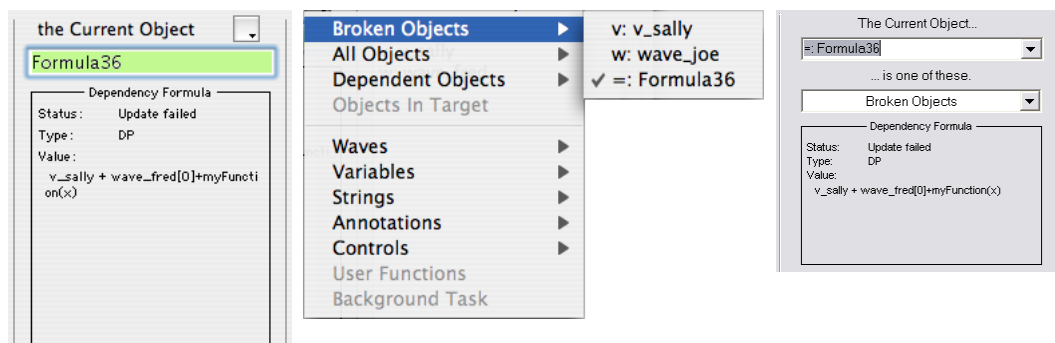


The functions have *not* vanished; Igor just can't display information about functions unless all the procedure windows compile successfully. When you fix the error, the user-defined functions will return.

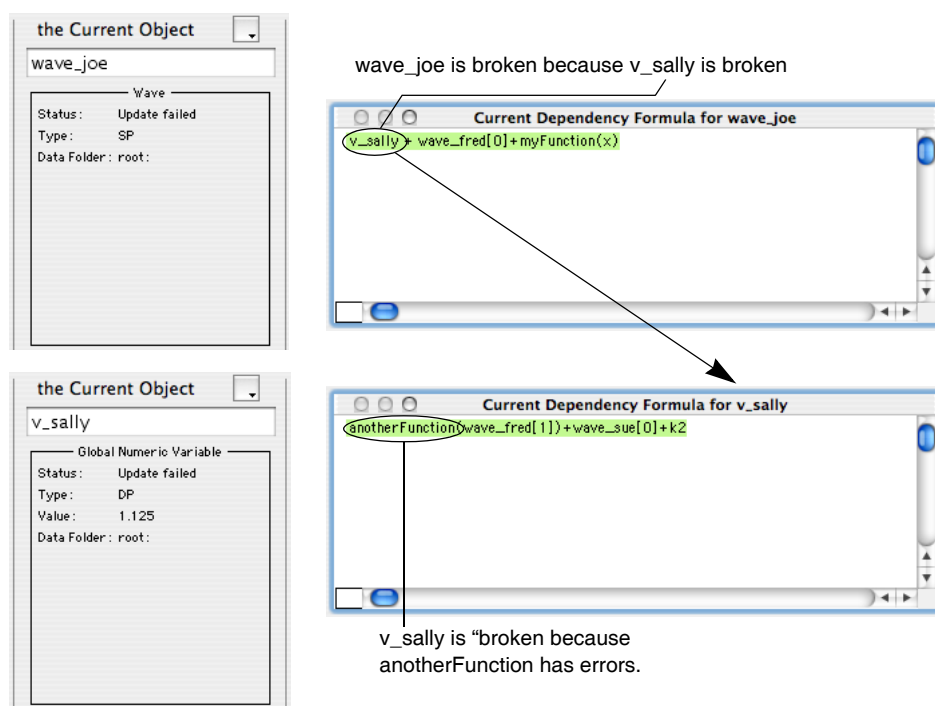
For this reason, it really makes more sense for you to click the Edit Procedure button in the error dialog. This brings up the procedure window in which the function is defined, but it does exit the Object Status dialog first (which would be devoid of function information, anyway). You should fix the error, and then reselect the Object Status dialog. You will find that the function is still the current object, and that all is well.

Broken Objects

If you *don't* fix errors in the user-defined function text, objects that reference the erroneous function will be "**broken**". Those objects are listed in the Broken Objects submenu. The Status field will show why the object is broken.



The usual causes for "Update failed" are either a syntax error in the dependency formula, or some object that the current object depends on (either directly or indirectly) has an error, has been renamed, or no longer exists. In this example, it is the unresolved error in the function `anotherFunction` (see **User Functions** on page III-419) which prevents `v_sally` from updating, which in turn causes the update of `wave_joe` through `Formula36` to fail:



Graphics Technology

As of version 6.1, Igor Pro uses more modern graphics code for drawing graphs, tables and page layouts. On Macintosh this involves the radical change of using Apple's Quartz routines rather than the ancient QuickDraw routines. On Windows just slightly more advanced code is used with a small amount of GDI+ instead of GDI.

The new graphics code supports arbitrary rotation for text and imported pictures and, on Macintosh, support for imported PDF pictures which can be placed in graphs, page layouts and formatted notebooks.

The new code does not support the old Macintosh PICT format or the old Windows WMF format. If you need to export PICT or WMF, you can make Igor use the old graphics code by executing this:

```
SetIgorOption UseOldGraphics=1
```

When this command is executed, all graphs and page layout windows are redrawn using the old code.

Windows users may need to revert to the old graphics when exporting EMF to certain programs that do not support the advanced GDI mode. In particular, users have reported Corel products as not accepting advanced GDI.

You can also turn the old graphics code on if you have a problem with the new code. In this case, please let us know why you needed to do that so we can address the problem.

Pictures

Igor can import pictures from other programs for display in graphs, page layouts and notebooks. It can also export pictures from graphs, page layouts and tables for use in other programs. Exporting is discussed in Chapter III-5, **Exporting Graphics (Macintosh)**, and Chapter III-6, **Exporting Graphics (Windows)**. This section discusses how you can import pictures into Igor, what you can do with them and how Igor stores them.

For information on importing images as data rather than as graphics, see **Loading Other Files** on page II-167.

Importing Pictures

There are three ways to import a picture.

- Pasting from the Clipboard into a graph, layout, or notebook
- Using the Pictures dialog (Misc menu) to import a picture from a file or from the Clipboard
- Using the **LoadPICT** operation (see page V-347) to import a picture from a file or from the Clipboard

Each of these methods, except for pasting into a notebook, creates a *named*, global picture object that you can use in one or more graphs or layouts. Pasting into a notebook creates a picture that is local to the notebook.

This table shows the types of graphics formats from which Igor can import pictures:

Format	How To Import	Notes
PICT	Paste or use Misc→Pictures or LoadPICT	Macintosh only
PDF	Paste or use Misc→Pictures or LoadPICT	Macintosh only
EMF (Enhanced Metafile)	Paste or use Misc→Pictures or LoadPICT	Windows only
BMP (Windows bitmap)	Use Misc→Pictures or LoadPICT	Windows only BMP is sometimes called DIB (Device Independent Bitmap).
PNG (Portable Network Graphics)	Use Misc→Pictures or LoadPICT	Cross-platform bitmap format
JPEG	Use Misc→Pictures or LoadPICT	Cross-platform bitmap format
TIFF (Tagged Image File Format)	Use Misc→Pictures or LoadPICT	Cross-platform bitmap format
EPS (Encapsulated PostScript)	Use Misc→Pictures or LoadPICT	High resolution vector format. Requires PostScript printer. On Windows, a screen preview is displayed on screen.

EPS files usually include a screen preview. The screen preview format is PICT on Macintosh and TIFF on Windows. The screen preview is not used or needed on Macintosh as the OS converts and displays EPS as PDF on the fly.

The platform-dependent formats (PDF, PICT, EMF, BMP) are drawn as gray boxes when displayed on the nonnative format. The cross-platform formats can be displayed on either platform, except that Macintosh EPS pictures display as boxes on Windows, because the screen preview format (PICT) is platform-dependent. Also, EPS pictures which have no screen preview display as boxes on Windows.

EPS pictures provide the highest quality but on Windows should only be used if you are certain you will be printing on a PostScript printer or exporting to a PostScript-savvy application such as Adobe Illustrator. On Macintosh, they can be used wherever a PDF could be used.

When non-EPS pictures are used in a graphic that is exported as EPS, the picture is rendered as a bitmap which has lower quality than an EPS picture.

See also **Picture Compatibility** on page III-395 for a discussion of Macintosh graphics on Windows and vice-versa.

The Picture Collection Stores Named Pictures

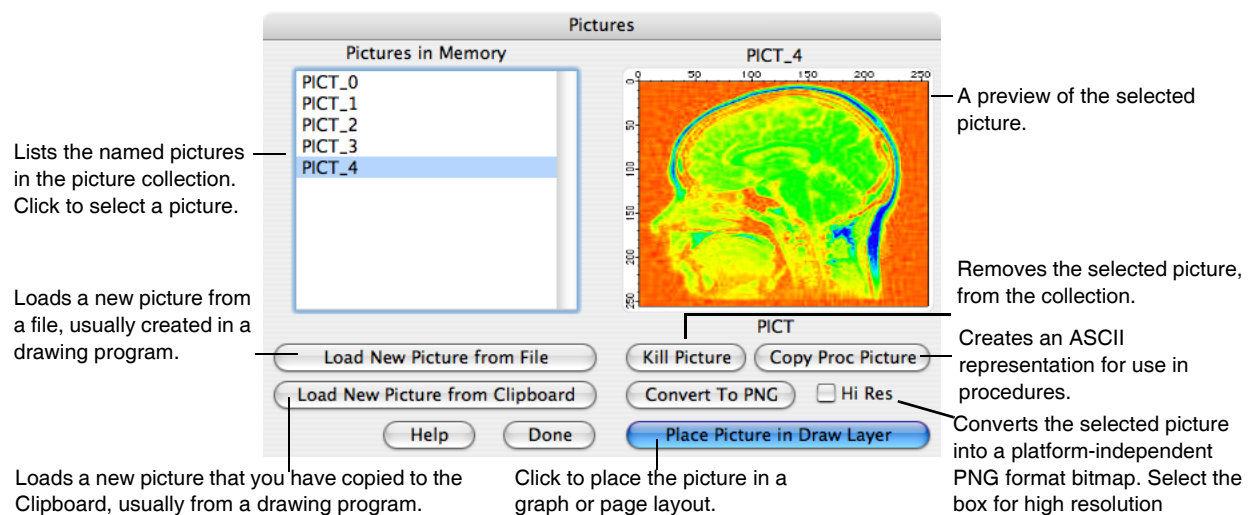
When you create a named picture using one of the techniques listed above, Igor stores it in the current experiment's **picture collection**. When you save the experiment, the picture collection is stored in the experiment file. You can inspect the collection using the Pictures dialog via the Misc menu.

Igor gives names to pictures so they can be referenced from an Igor procedure. For example, if you paste a picture into a layout, Igor assigns it a name of the form "PICT_0" and stores it in the picture collection. If you then close the layout and ask Igor to create a recreation macro, the macro will reference the picture by name.

You can rename a named picture using the Pictures dialog in the Misc menu, the Rename Objects dialog in the Misc menu, the Rename dialog in the Data menu, or the **RenamePICT** operation (see page V-520). You can kill a named picture using the Pictures dialog or the **KillPICTs** operation (see page V-322).

Pictures Dialog

The Pictures dialog permits you to view the picture collection, to add pictures, to remove pictures and to place a picture into a graph or page layout. It also can place a copy of a picture into a formatted notebook. To invoke it, choose Pictures from the Misc menu.



The Kill This Picture button will be dimmed if the selected picture is used in a currently open graph or layout.

Note: Igor determines if a picture is in use by checking to see if it is used in an *open* graph or layout window.

If you kill a graph or layout that contains a picture and create a recreation macro, the recreation macro will *refer* to the picture by name. However, Igor does not check for this. It will consider the picture to be unused and will allow you to kill it. If you later run the recreation macro, an error will occur when the macro attempts to append the picture to the graph or layout. Therefore, don't kill a picture unless you are sure that it is not needed.

Notebook Pictures

When you paste a picture into a formatted notebook, you create a notebook picture. These work just like pictures in a word processor document. You can copy and paste them. These pictures will not appear in the Pictures or Rename Objects dialogs.

Igor Extensions

Igor includes a feature that allows a C or C++ programmer to extend its capabilities. An Igor extension is called an "XOP" (short for "external operation"). The term XOP comes from that fact that, originally, adding a command line operation was all that an extension could do. Now extensions can add operations, functions, menus, dialogs and windows.

WaveMetrics XOPs

The "Igor Pro Folder/Igor Extensions" and "Igor Pro Folder/More Extensions" folders contain XOPs that we developed at WaveMetrics. These add capabilities such as file-loading and instrument control to Igor and

also serve as examples of what XOPs can do. These XOPs range from very simple to rather elaborate. Most XOPs come with help files that describe their operation.

The WaveMetrics XOPs are described in the XOP Index help file, accessible through the Windows→Help Windows submenu.

Third Party Extensions

A number of Igor users have written XOPs to customize Igor for their particular fields. Some of these are freeware, some are shareware and some are commercial programs. WaveMetrics publicizes third party XOPs through our Web page. User-developed XOPs are available from <http://www.igorexchange.com>. Also, we make some third party XOPs available via FTP. See **FTP Sites** on page II-15 for details on FTP.

Activating Extensions

The Igor installer creates a folder called Igor Extensions inside the Igor Pro Folder. The Igor installer puts some XOP files in this folder. XOPs in this folder are loaded when Igor starts up.

The installer puts less-frequently used XOPs in "Igor Pro Folder/More Extensions". These are not available unless you activate them.

If you place in "Igor Pro Folder/Igor Extensions" an alias (Macintosh) or shortcut (Windows) for an XOP file or a folder containing XOP files, Igor loads those files also. However, this is not the recommended way to activate an XOP.

Your operating system may not allow you to make changes to the contents of your Igor Pro Folder, for example, if you do not have permission to write to that folder. For that reason Igor Pro 6.1 or later also loads extensions from another folder - "Igor Pro User Files/Igor Extensions" (see **Igor Pro User Files** on page II-46 for details). You can locate this folder by selecting Help→Show Igor Pro User Files.

If you press the shift key while clicking the Help menu, you can choose Help→Show Igor Pro Folder and User Files. Igor then opens the Igor Pro Folder and the Igor Pro User Files folder on the desktop, making it easy to drag aliases/shortcuts into "Igor Pro User Files/Igor Extensions". This is the recommended way to activate additional XOPs for a given user.

If you want to activate an extension for all users on a given machine, you can put the alias or shortcut in "Igor Pro Folder/Igor Extensions" folder. Windows 7 does not permit you to manually create a shortcut in the Igor Pro Folder but you can create the shortcut on the desktop and drag it into the Igor Pro Folder.

Changes that you make to either Igor Extensions folder take effect the next time Igor is launched.

See **Creating Igor Extensions** on page IV-181 if you are a programmer interested in writing your own XOPs.

XOPs on Intel Macintosh

To run an XOP on Intel Macintosh it must be ported to the Intel architecture. WaveMetrics has ported most of its XOPs and distributes them as universal executables, which means that they run on both PowerPC and Intel Macintosh. The Igor Pro 6 application is also universal.

Because third-party libraries are not yet available, the following WaveMetrics XOPs have not yet been ported to run with the Intel version of Igor Pro: NIGPIB, NIGPIB2.

In addition, the NetCDF Loader XOP, written by Igor user Katsuhisa Kitano and ported to Mac OS X by WaveMetrics, has not yet been ported to Intel.

If you use a third-party XOP that has not yet been ported to Intel, you can not run it with the Intel version of Igor.

The Igor Pro installer places non-Intel WaveMetrics Macintosh XOPs in the PPC Extensions folder.

If you run the Intel Macintosh version of Igor Pro with a PowerPC-only XOP installed, it will not display an error message and will ignore the PowerPC-only XOP. Similarly, if you run the PowerPC version of Igor with an Intel-only XOP installed, it will not display an error message and will ignore the Intel-only XOP.

Running PowerPC XOPs on Intel Macintosh

If you need to run an XOP that has not been ported to Intel Macintosh, you must run the PowerPC version of Igor Pro. You can do this on an Intel Macintosh by checking the Open Using Rosetta checkbox in the Finder Info window for the Igor Pro application located in the Igor Pro folder.

With the Open Using Rosetta checkbox checked, the PowerPC code in the Igor Pro 6 application package will run any PowerPC XOPs that you invoke.

Memory Management

On Mac OS X, each 32-bit application (such as Igor Pro) is allocated 4 GB of virtual address space. On Windows, 32-bit applications are allocated 2 GB of virtual address space by default but there is a technique (explained below) for increasing this to 3 GB or 4 GB.

Theoretically, applications can allocate memory up the limit of the virtual address space. The operating system allocates physical memory as necessary. If the total amount of allocated virtual memory exceeds the total amount of physical memory, the operating system will temporarily write some data to disk and load other data into physical memory — a process called “swapping”. This can make the computer very slow.

When a program is launched, the operating system maps the program’s code into its virtual address space, thereby diminishing the space available for data. During the program’s initialization, it allocates many blocks of data, thousands in Igor’s case, for things like windows, controls and internally-used structures. This further reduces the amount of virtual memory space available for data.

As you work, creating windows, procedures, variables, waves and other objects, Igor allocates more blocks of virtual memory. When you kill a window, wave or other object, Igor deallocates blocks of virtual memory. This leaves free blocks in the virtual memory space.

Blocks of virtual memory space that are free because of deallocation are generally discontinuous. The largest contiguous block is smaller than the total amount of free virtual memory. The free virtual memory space is split into fragments that can not be joined because they are separated by allocated blocks. As you allocate and deallocate memory, the virtual memory space becomes more and more fragmented.

Another cause of fragmentation is the loading of dynamic link libraries.

When you create an object, for example, a wave, Igor needs a contiguous block of virtual memory. If the space needed for the wave is larger than the largest contiguous block of virtual memory, the allocation will fail and Igor will return an out-of-memory error.

Fragmentation sets the limit for the largest wave you can create. This is an issue if you are creating very large waves that require, typically, 250 MB or more. If you are creating a large number of smaller waves, for example, 100 waves of 10 MB each, fragmentation is generally not an issue.

Increasing Virtual Memory Space in Windows VISTA and Windows 7

If you are running Windows VISTA x64 (64-bit) or Windows 7 x64 (64-bit), the operating system allows 32-bit applications like Igor to use 4 GB of virtual address space. You can not increase this.

On Windows 7 x64 with 4 GB of physical memory the largest wave we were able to create was 2000 MB.

If you are running Windows VISTA (32-bit) or Windows 7 (32-bit), by default applications get a 2 GB virtual address space. You can increase this to 3 GB using the BCDEdit program to change your boot settings. To do this, click the Windows Start button, choose Programs→Accessories and then right-click Command Prompt. Choose Run As Administrator. In the Command Prompt window, enter this command:

```
BCDEdit /Set IncreaseUserVa 3072
```

Now reboot. Igor should now have a 3 GB virtual address space.

Increasing Virtual Memory Space in Windows XP

If you are running Windows XP Professional x64 (64-bit), the operating system allows 32-bit applications like Igor to use 4 GB of virtual address space. You can not increase this.

If you are running Windows XP Professional (32-bit), by default applications get a 2 GB virtual address space. You can increase this to 3 GB by changing a setting on your computer, specifically by adding the /3GB flag to your C:\boot.ini file.

Note: Be careful! If you mess up your boot.ini file, your computer will not boot. This procedure should be attempted only by advanced Windows computer users.

The boot.ini is invisible by default. To see it you must use Explorer's Tools→Folder Options to enable viewing hidden files and disable hiding protected system files. Alternatively you can use the System control panel, Advanced Tab, Startup and Recovery section to edit it.

Back up the boot.ini file before making any changes to it.

Open boot.ini in Notepad and add the /3GB flag to the appropriate partition. For example, after adding the flag, the relevant line in boot.ini may look like this:

```
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"  
/fastdetect /3GB
```

When you next boot, the operating system will allow Igor to use 3 GB of virtual address space instead of 2 GB.

The /3GB flag has no effect on programs that do not have the special flag set.

On Windows XP SP3 with 2 GB of physical memory and a 3 GB virtual address space, the largest wave we were able to create was about 600 MB. We were able to increase this to 880 MB by applying a Microsoft hotfix that reduces memory fragmentation. The hotfix is available from <http://support.microsoft.com/kb/894472>.

Macintosh System Requirements

Igor Pro 6.1 requires Mac OS X 10.4.0 or later. It runs native on PowerPC and Intel processors.

Igor Pro 6 does not run on Mac OS 9.

Windows System Requirements

Igor Pro requires Windows XP, Windows Vista or Windows 7.

Igor Pro 6.1 does not run on Windows 95, Windows 98, Windows ME, Windows NT or Windows 2000.

Crashes

A crash results from a software bug and prevents a program from continuing to run. Crashes are highly annoying at best and, at worst, can cause you to lose valuable work.

WaveMetrics uses careful programming practices and extensive testing to make Igor as reliable and bug-free as we can. However in Igor as in any complex piece of software it is impossible to exterminate all bugs. Also, crashes can sometimes occur in Igor because of bugs in other software, such as printer drivers, video drivers or system extensions.

We are committed to keeping Igor a solid and reliable program. If you experience a crash, we would like to know about it.

When reporting a crash to WaveMetrics, please include the following information:

- The exact version of Igor Pro (e.g., 6.10) that you are running.
- The exact operating system (e.g., Mac OS X 10.4.7, Windows XP) that you are running.
- A description of what actions preceded the crash and whether it is reproducible.
- A recipe for reproducing the crash, if possible.
- A crash log (described below), if possible.

We have three methods for determining the cause of a crash.

The first method is to reproduce it on our computers. For this we need a recipe from you, if possible.

If we can not reproduce the crash, our second method is to examine the source code looking for possible bugs. For this, we need a detailed description from you of what you were doing when the crash occurred so that we know what part of the source code to examine.

Our last resort is to examine the crash log for a clue as to where the crash occurred. This sometimes provides useful information, sometimes not.

Crash Logs on Mac OS X

When a crash occurs on Mac OS X, most of the time the system is able to generate a crash log. You can usually find it at:

```
/Users/<user>/Library/Logs/DiagnosticReports/Igor Pro_<date>_<machinename>.crash
```

or

```
/Users/<user>/Library/Logs/CrashReporter/Igor Pro.crash.log
```

where <user> is your user name.

For irreproducible crashes, send this log as an attachment to support@wavemetrics.com. Include the other information listed above.

Crashes On Windows

When a crash occurs on Windows XP, but not Windows VISTA or Windows 7, most of the time the system is able to generate a crash log. You can usually find it at:

```
C:\Documents and Settings\All Users\Application Data\Microsoft\Dr Watson\drwtsn32.log
```

The Application Data folder is normally hidden. To make it visible, go into the All Users folder and choose Tools→Folder Options. Click the View tab, then the Show Hidden Files and Folders radio button, and the OK button.

You can also search for the drwtsn32.log file. In this case, make sure to include hidden files and folders in the search.

For irreproducible crashes, send this log as an attachment to support@wavemetrics.com. Include the other information listed above.

If you install a software development system such as Microsoft Visual C++, the development system will set itself up as the default debugger and Dr. Watson will not run when a crash occurs. In this case you will have no drwtsn32.log file.

On Windows VISTA and Windows 7, the operating system does not produce a user-accessible crash log. Your best bet is to try to determine a recipe to reproduce the crash or a pattern that leads to the crash and send a report to support@wavemetrics.com.

