Chapter
IV-8

Debugging

Debugging Procedures

There are two techniques for debugging procedures in Igor:

- Using print statements
- Using the symbolic debugger

For most situations, the symbolic debugger is the most effective tool. In some cases, a strategically placed print statement is sufficient.

Debugging With Print Statements

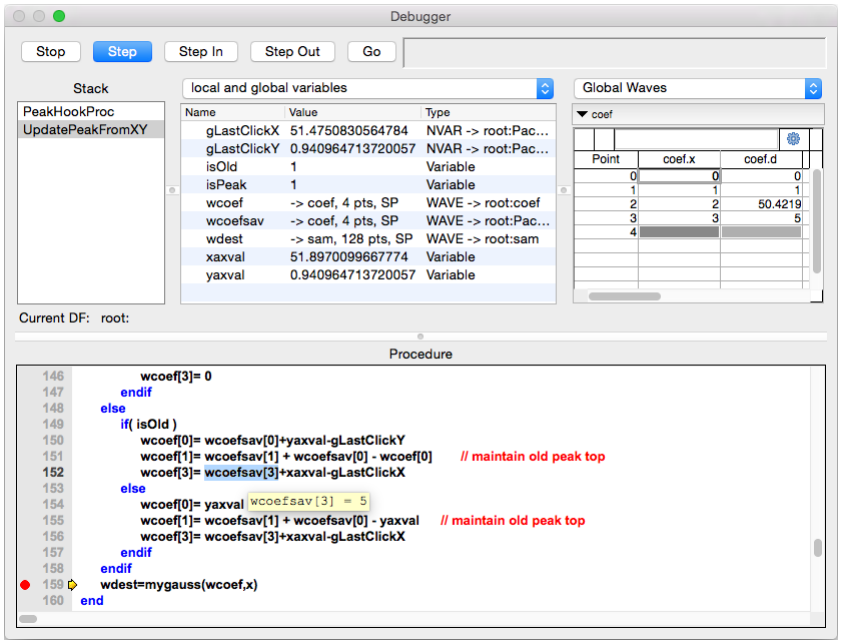
This technique involves putting print statements at a certain point in a procedure to display debugging messages in Igor's history area. In this example, we use Printf to display the value of parameters to a function and then Print to display the function result.

```
Function Test(w, num, str)
   Wave w
   Variable num
   String str

   Printf "Wave=%s, num=%g, str=%s\r", NameOfWave(w), num, str

   <body of function>

   Print result
   return result
End
```

See **Creating Formatted Text** on page IV-244 for details on the Printf operation.

The Debugger

When a procedure doesn't produce the results you want, you can use Igor's built-in debugger to observe the execution of macros and user-defined functions while single-stepping through the lines of code.



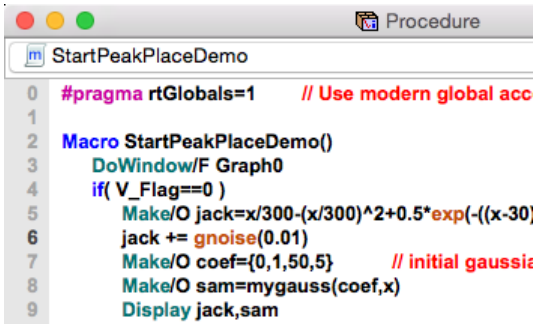
The debugger is normally disabled. Select Enable Debugger in either the Procedure menu or the contextual menu shown by control-clicking (Macintosh) or by right-clicking (Windows) in any procedure window.

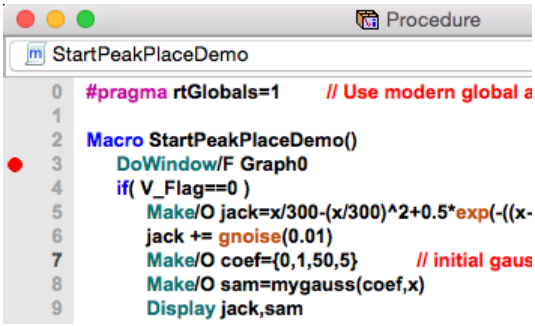
Igor displays the debugger window when one of the following events occurs:

1. A breakpoint that you previously set is hit
2. An error occurs, and you have enabled debugging on that kind of error
3. An error dialog is presented, and you click the Debug button
4. The **Debugger** command is executed

Setting Breakpoints

When you want to observe a particular routine in action, set a breakpoint on the line where you want the debugger to appear. To do this, open the procedure window which contains the routine, and click in the left "breakpoint margin". The breakpoint margin appears only if the debugger has been enabled. These graphics show the procedure windowwith the debugger disabled (left) and enabled (right):





The red dot denotes a breakpoint that you have set.

When a line of code marked with a breakpoint is about to execute, Igor displays the debugger window.

Click the red dot again to clear the breakpoint. Control-click (*Macintosh*) or right-click (*Windows*) and use the pop-up menu to clear all breakpoints or disable a breakpoint on the currently selected line of the procedure window.

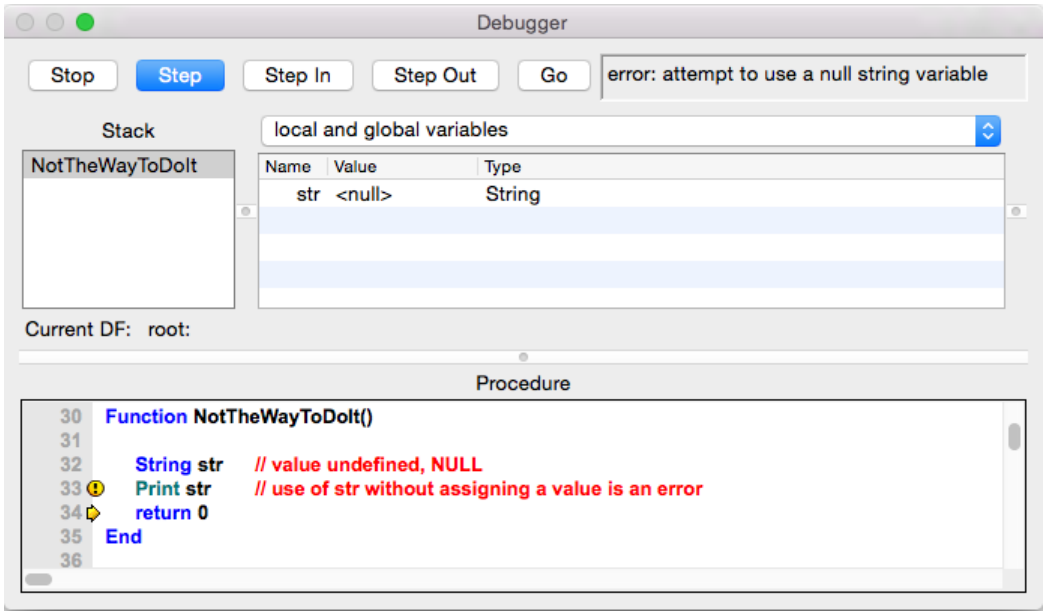
Debugging on Error

You can automatically open the debugger window when an error occurs. There are two categories of errors to choose from:

Debug On Error	Any runtime error except failed NVAR, SVAR, or WAVE references.
NVAR SVAR WAVE Checking	Failed NVAR, SVAR, or WAVE references.

We recommend that Igor programmers turn both of these options on to get timely information about errors.

Use the Procedure or contextual menus to enable or disable both error categories. If the selected error occurs, Igor displays the debugger with an error message in its status area. The error message was generated by the command indicated by a round yellow icon, in this example the Print str command:



Sometimes you do something that you know may cause an error and you want to handle the error yourself, without breaking into the debugger. One such case is attempting to access a wave or variable that may or may not exist. You want to test its existence without breaking into the debugger.

You can use the /Z flag to prevent the Debug on Error feature from kicking in when an NVAR, SVAR, or WAVE reference fails. For example:

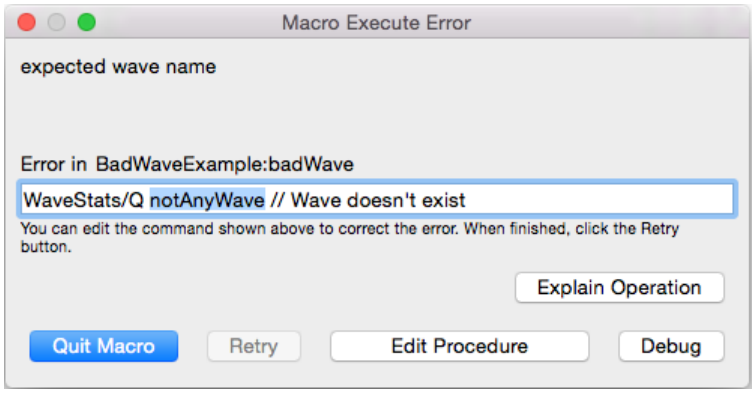
```
WAVE/Z w = <path to possibly missing wave>
if (WaveExists(w))
   <do something with w>
endif
```

In other cases where an error may occur and you want to handle it yourself, you need to temporarily disable the debugger and use **GetRTError** to get and clear the error. For example:

```
Function DemoDisablingDebugger()
   DebuggerOptions                  // Sets V_enable to 1 if debugger is enabled
   Variable debuggerEnabled=V_enable
   DebuggerOptions enable=0         // Disable debugger

   String name = ";"                // This is an illegal wave name
   Make/O $name                     // So this will generate and error

   DebuggerOptions enable=debuggerEnabled    // Restore

   Variable err = GetRTError(1)              // Clear error
   if (err != 0)
      Printf "Error %d\r", err
   else
      Print "No error"
   endif
End
```

Macro Execute Error: The Debug Button

When the debugger is enabled and an error occurs in a macro, Igor presents an error dialog with, in most cases, a Debug button. Click the Debug button to open the debugger window.



Errors in macros (or procs) are reported immediately after they occur.

When an error occurs in a user-defined function, Igor displays an error dialog long after the error actually occurred. The Debug On Error option is for programmers and displays errors in functions when they occur.

Stepping Through Your Code

Single-stepping through code is useful when you are not sure what path it is taking or how variables wound up containing their values.

Begin by enabling the debugger and setting a breakpoint on the line of code you are interested in, or begin when the debugger automatically opens because of an error. Use the buttons at the top of the debugger window to step through your code:

The Stop Button

The Stop button ceases execution of the running function or macro before it completes. This is equivalent to clicking Igor's Abort button while the procedure is running.

Keyboard shortcuts: (none)

Pressing Command-period on a Macintosh while the debugger window is showing is equivalent to clicking the *Go* button, not the Stop button.

The Step Button

The Step button executes the next line. If the line contains a call to one or more subroutines, execution continues until the subroutines return or until an error or breakpoint is encountered. Upon return, execution halts until you click a different button.

Keyboard shortcuts: Enter, keypad Enter, or Return

The Step Into Button

The Step Into button executes the next line. If that line contains a call to one or more subroutines, execution halts when the first subroutine is entered. The Stack list of currently executing routines shows the most recently entered routine as the last item in the list.

Keyboard shortcuts: +, =, or keyPad +

The Step Out Button

The Step Out button executes until the current subroutine is exited, or an error or breakpoint is encountered.

Keyboard shortcuts: -, _ (underscore) or keypad -

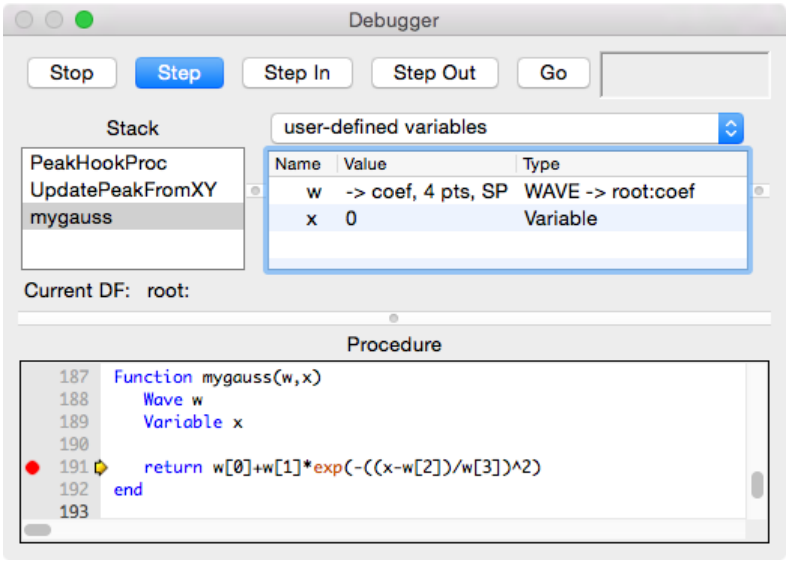
The Go Button

The Go button resumes program execution. The debugger window remains open until execution completes or an error or breakpoint is encountered.

Keyboard shortcuts: Esc

The Stack and Variables Lists

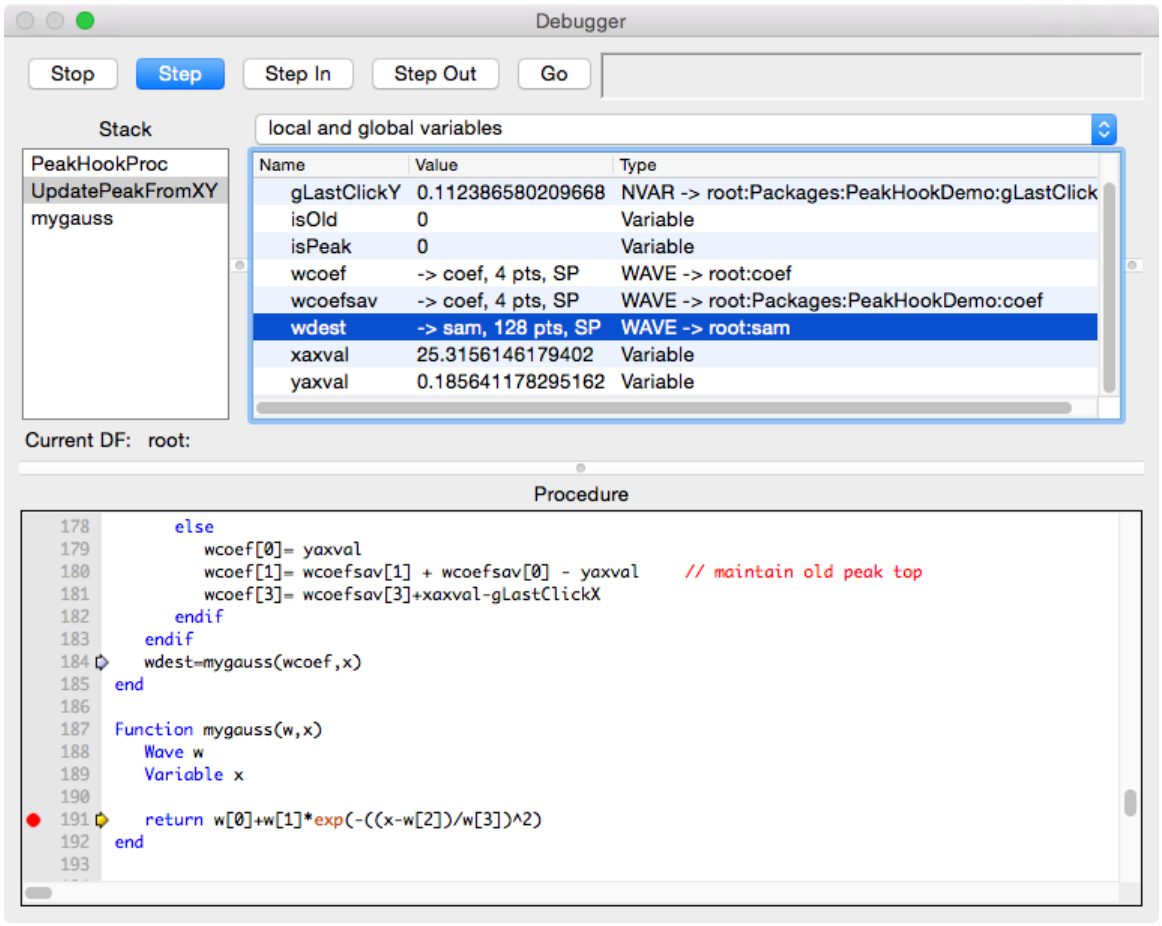
The Stack List shows the routine that is currently executing and the chain of routines that called it. The top item in the list is the routine that began execution and the bottom item is the routine which is currently executing.



In this example, the routine that started execution is PeakHookProc, which most recently called UpdatePeakFromXY, which then called the currently executing mygauss user function.

The Variables List, to the right of the Stack List, shows that the function parameters w and x have the values coef (a wave) and 0 (a number). The pop-up menu controls which variables are displayed in the list; the example shows only user-defined local variables.

You can examine the variables associated with any routine in the Stack List by simply selecting the routine:



Here we've selected UpdatePeakFromXY, the routine that called mygauss (see the light blue arrow). Notice that the Variables List is showing the variables that are local to UpdatePeakFromXY.

For illustration purposes, the Variables List has been resized by dragging the dividing line, and the pop-up menu has been set to show local and global variables and type information.

The Variables List Columns

The Variables List shows either two or three columns, depending on whether the "show variable types" item in the Variable pop-up menu is checked.

Double-clicking a column's header resizes the column to fit the contents. Double-clicking again resizes the column to a default width.

The first column is the name of the local variable. The name of an NVAR, SVAR, or WAVE reference is a name local to the macro or function that refers to a global object in a data folder.

The second column is the value of the local variable. Double-click the second column to edit numbers in-place, double-click anywhere on the row to "inspect" waves, strings, SVARS, or char arrays in structures in the appropriate Inspector.

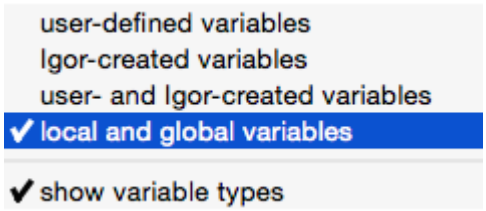
In the case of a wave, the size and precision of the wave are shown here. The "->" characters mean "refers to". In our example wcoef is a local name that refers to a (global) wave named coef, which is one-dimensional, has 4 points, and is single precision.

To determine the value of a particular wave element, use an inspector as described under **Inspecting Waves**.

The optional third column shows what the type of the variable is, whether Variable, String, NVAR, SVAR, WAVE, etc. For global references, the full data folder path to the global is shown.

Variables Pop-Up Menu

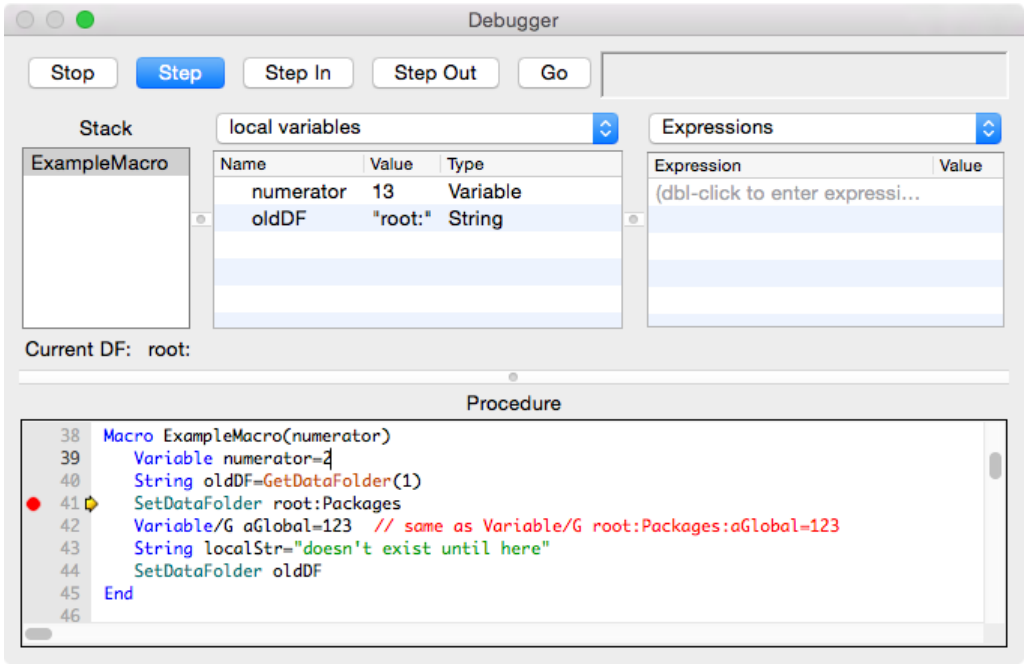
The Variables pop-up menu controls which information is displayed in the Variables List. When debugging a function, it looks like this:



When debugging a macro, proc or window macro, the first two items in the popup menu are unavailable.

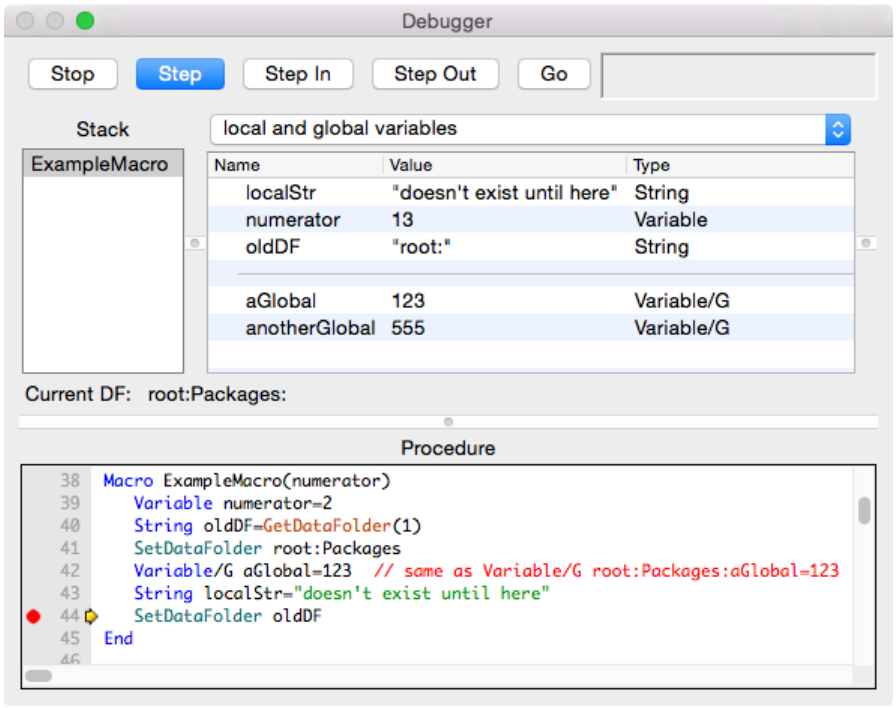
Macro Variables

The ExampleMacro below illustrates how variables in Macros, Procs or Window procedures are classified as locals or globals:



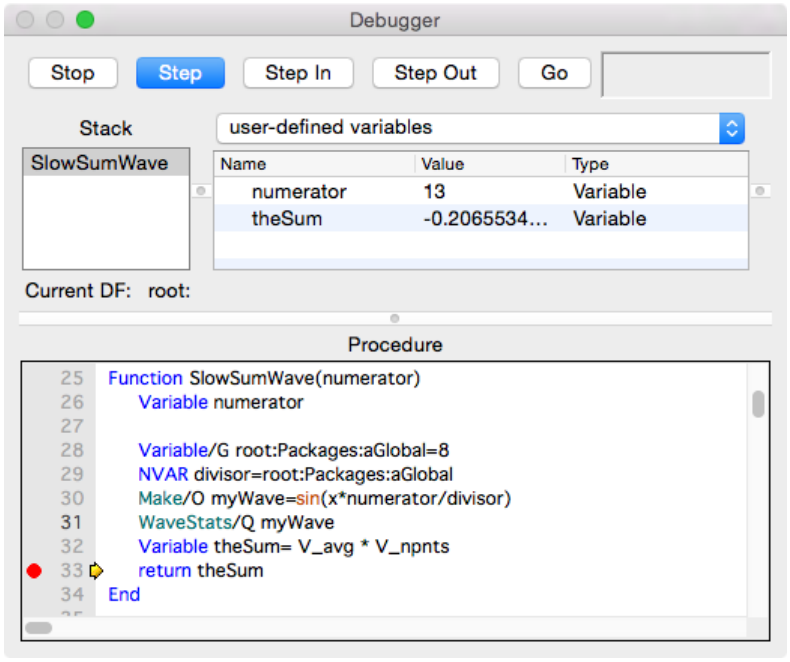
Local variables in macros include all items passed as parameters (numerator in this example) and local variables and local strings (oldDF) whose definitions have been executed, and Igor-created local variables created by operations such as WaveStats after the operation has been executed. Note that localStr isn't listed, because the command has not yet executed.

Global variables in macros include all items in the current data folder, whether they are used in the macro or not. If the data folder changes because of a SetDataFolder operation, the list of global variables also changes. Note that there are no NVAR, SVAR, WAVE or STRUCT references in a macro.



Function Variables

The SlowSumWaveFunction example below illustrates how different kinds of variables in functions are classified:

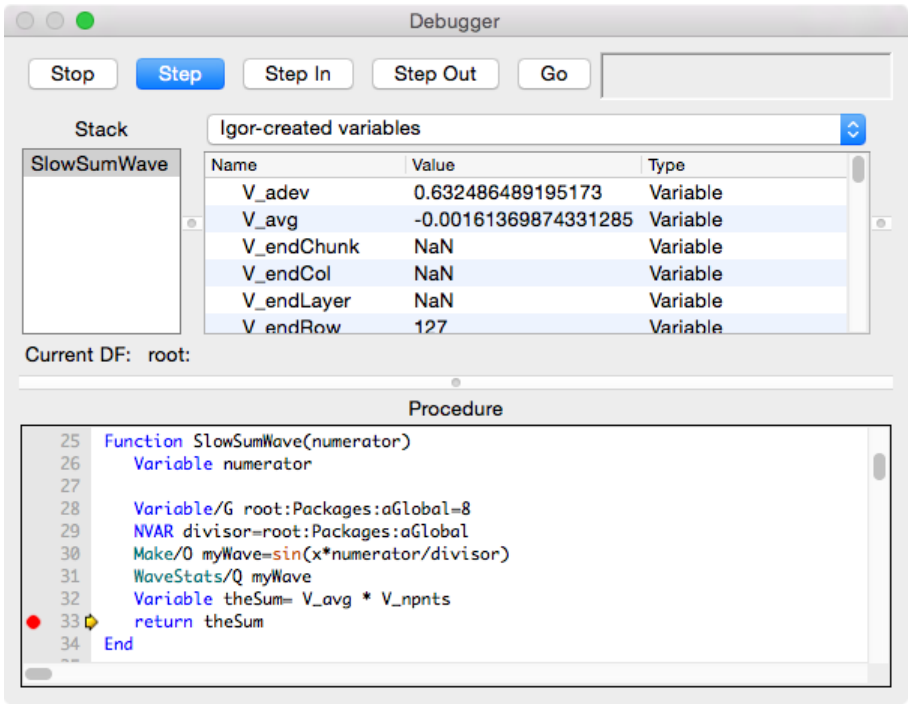


User-defined variables in functions include all items passed as parameters (numerator in this example) and any local strings and variables.

Local variables exist while a procedure is running, and cease to exist when the procedure returns; they never exist in a data folder like globals do.

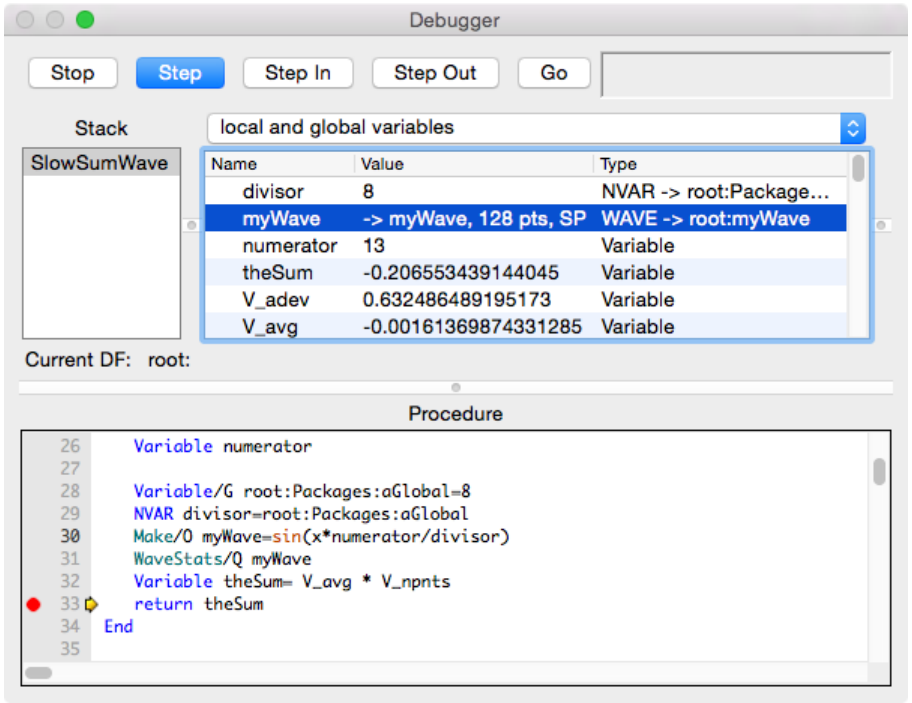
NVAR, SVAR, WAVE, Variable/G and String/G references point to global variables, and therefore, aren't listed as user-defined (local) variables.

Use "Igor-created variables" to show local variables that Igor creates for functions when they call an operation or function that returns results in specially-named variables. The **WaveStats** operation (see page V-934), for example, defines V_adev, V_avg, and other variables to contain the statistics results:



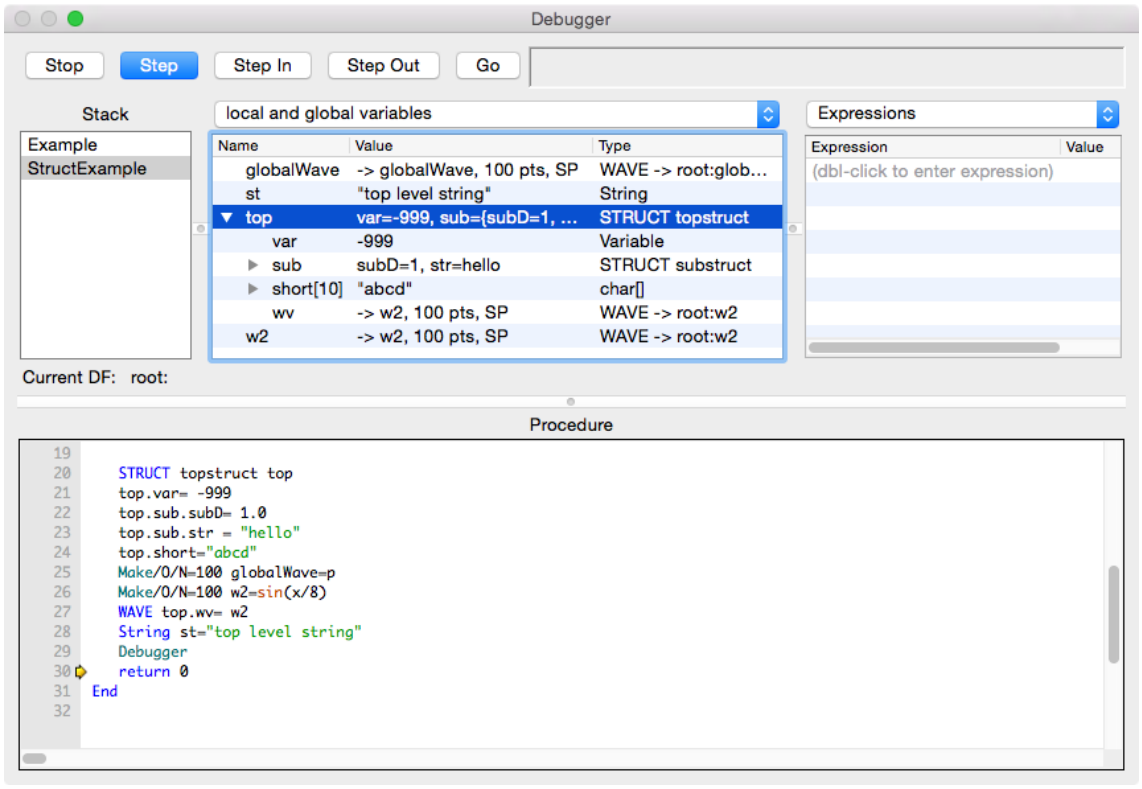
The "user- and Igor-created" menu item shows both kinds of local variables.

The "local and global variables" item shows user-created local variables, most Igor-created local variables, and references to global variables and waves through NVAR, SVAR, and WAVE references:

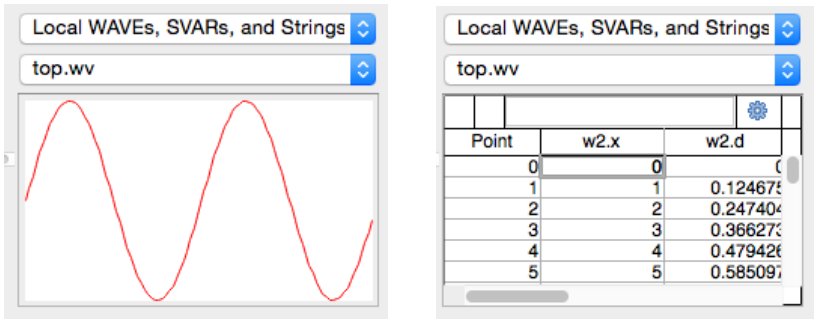


Function Structures

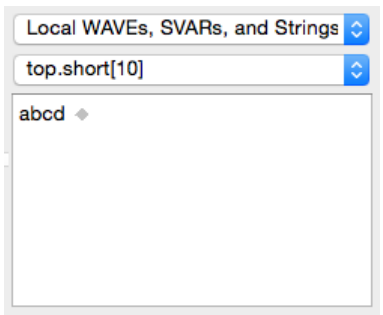
The elements of a structure (see **Structures in Functions** on page IV-91) are displayed in tree form in the Variables "list". Click the triangles to expand or collapse a node in the structure, or double-click the row:



Double-clicking a WAVE element (such as top.wv) will send it to the Wave Inspector (either a table or graph depending on what is checked in the Inspector popup):



Double-clicking a String element or char array (such as top.short) will send it to the String Inspector:



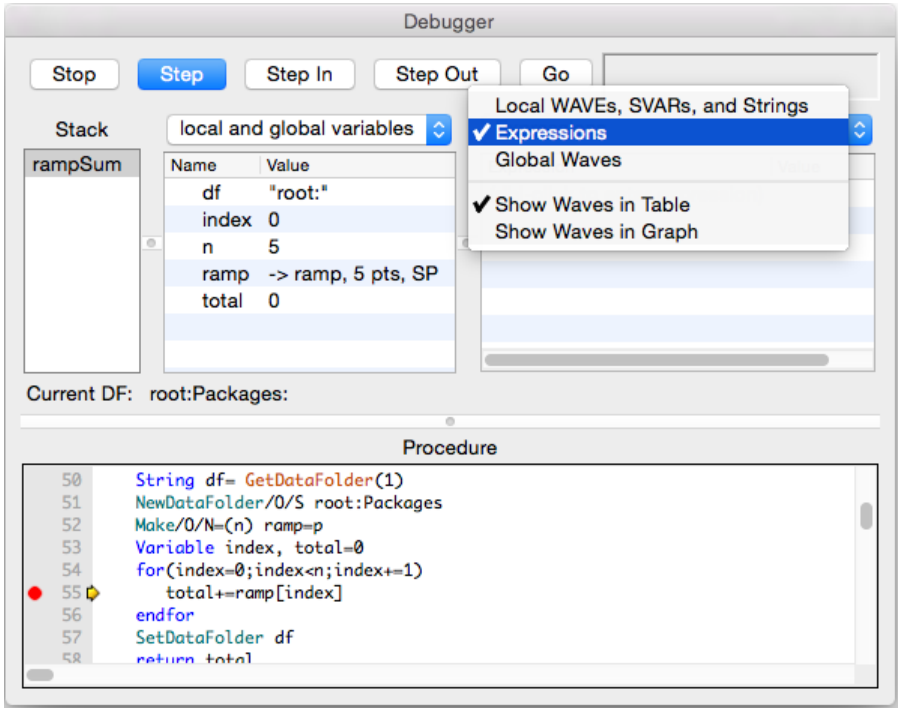
The Current Data Folder

The "Current DF" displays the path to the current data folder. You can select and copy the data folder path. See **Data Folders** on page II-99 for more information about data folders.

Graph, Table, String, and Expressions Inspectors

One of the Wave, String, or Expressions Inspectors is visible to right side of the variables list. This pane is hidden when the divider between the pane and the variables list is dragged all the way to the right. Drag the divider to the left to show the pane. You may need to widen the window to make room.

Use the popup to select the Inspector you want.

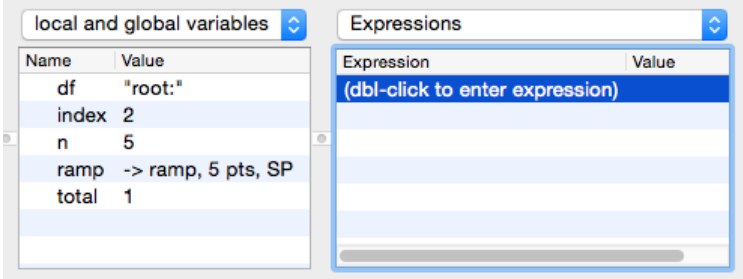


The pop-up menu controls what is shown in the pane:

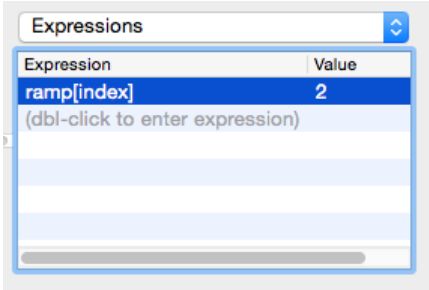
Pop-up Menu Selection	Pane Contents
Local WAVEs, SVARs, and Strings	A list of these things in the selected function. References to free waves are also listed here, as are elements in a STRUCT that are "inspectable", including char arrays which can be viewed as if they were Strings.
Local Strings	A list of Strings local to the selected function, macro or proc.
Expressions	Numeric or string expressions which are evaluated in the context of the selected function or macro.
Global Waves	A popup wave selector to display any wave in a global data folder. Free waves are not listed here.
Show Waves in Table Show Waves in Graph	Waves will be displayed in a table or graph, depending on which one of these two is checked.

Expressions Inspector

Selecting "Expressions" from the inspector popup shows a list of Expressions and their values:



Replace the "(dbl-click to enter expression)" invitation by clicking it, typing a numeric or string expression, and pressing Return.



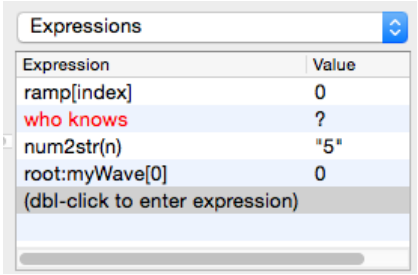
Adding an expression adds a blank row at the end of the list that can be double-clicked to enter another expression. You can edit any of the expressions by double-clicking and typing.

The expression can be removed by selecting it and pressing Delete or Backspace.

The result of the expression is recomputed when stepping through procedures. The expressions are evaluated in the context of the currently selected procedure.

Global expressions are evaluated in the context of the current data folder, though you can specify the data folder explicitly as in the example below.

If an expression is invalid the result is shown as “?” and the expression is changed to red:



The expressions are discarded when a new Igor experiment is opened or when Igor quits.

Inspecting Waves

You can "inspect" (view) the contents of a wave in either a table or a graph. They aren't full-featured tables or graphs, as there are no supporting dialogs for them. You can change their properties using contextual menus.

Select the Wave to be inspected by one of three methods:

1. Choose "Global Waves", and then choose a wave from the popup wave browser.
2. Choose "Local WAVEs, SVARs, and Strings", and then choose a wave from among the objects listed.
3. Double-click any WAVE reference in the Variables list.

Inspecting Waves in a Table

You can edit the values in a wave using the table, just like a regular table. With the contextual menu you can alter the column format, among other things.

Inspecting Waves in a Graph

You can view waves in a Graph. With the contextual menu, you can choose to show or hide the axes and change display modes.

Two-dimensional waves are displayed as an image plot.

Inspecting Strings

Select a String or char array to be inspected by two methods:

1. Choose "Local WAVEs, SVARs, and Strings", and then choose a String, SVAR or char array from among the objects listed.
2. Double-click any String, SVAR or char array in the Variables list.

The Procedure Pane

The procedure pane contains a copy of the procedure window of the routine selected in the Stack List. You can set and clear breakpoints in this pane just as you do in a procedure window, using the breakpoint margin and the Control-click (*Macintosh*) or right-click (*Windows*) menu.

A very useful feature of the debugger is the automatic text expression evaluator that shows the value of a variable or expression under the cursor. The value is displayed as a tooltip. This is often faster than scrolling through the Variables List or entering an expression in the Expressions List to determine the value of a variable, wave, or structure member reference.

The value of a variable can be displayed whether or not the variable name is selected. To evaluate an expression such as "wave[ii]+3", the expression must be selected and the cursor must be over the selection.

The debugger won't evaluate expressions that include calls to user-defined functions; this prevents unintended side effects (a function could overwrite a wave's contents, for example). You can remove this limitation by creating the global variable root:V_debugDangerously and setting it to 1.

After You Find a Bug

Editing in the debugger window is disabled because the code is currently executing. Tracking down the routine after you've exited the debugger is easy if you follow these steps:

1. Scroll the debugger text pane back to the name of the routine you want to modify, and select it.
2. Control-click (Macintosh) or Right-click (Windows) the name, and choose "Go to <routineName>" from the pop-up menu.
3. Exit the debugger by clicking the "Go" button or by pressing Escape.

Now the selected routine will be visible in the top procedure window, where you can edit it.

Debugger Shortcuts

Action	Shortcut
To enable debugger	Choose Enable Debugger from the Procedure menu or choose Enable Debugger from the procedure window's pop-up menu after Control-clicking (*Macintosh*) or right-clicking (*Windows*).
To automatically enter the debugger when an error occurs	Choose Debug on Error from the Procedure menu or choose Enable Debugger from a procedure window's pop-up menu after Control-clicking (*Macintosh*) or right-clicking (*Windows*).
To set or clear a breakpoint	Click in the left margin of the procedure window or click anywhere on the procedure window line where you want to set or clear the breakpoint and choose Set Breakpoint or Clear Breakpoint from a procedure window's pop-up menu after Control-clicking (*Macintosh*) or right-clicking (*Windows*).
To enable or disable a breakpoint	Shift-click a breakpoint in the left margin of the procedure window.
	Click anywhere on the procedure window line where you want to enable or disable the breakpoint and choose Enable Breakpoint or Disable Breakpoint procedure window's pop-up menu after Control-clicking (*Macintosh*) or right-clicking (*Windows*).
To execute the next command	On *Macintosh* press Enter, keypad Enter, or Return. For *Windows*, if no button has the focus, press Enter or Return. Otherwise, click the yellow arrow button.
To step into a subroutine	Press the +, =, or keypad + keys, or click the blue descending arrow button.
To step out of a subroutine to the calling routine	Press the -, _ (underscore) or keypad - keys, or click the blue ascending arrow button.
To resume executing normally	Press Escape (Esc), or click the green arrow button.
To cancel execution	Click the red stop sign button.
To edit the value of a macro or function variable	Double-click the second column of the variables list, edit the value, and press Return or Enter.
To set the value of a function's string to null	Double-click the second column of the variables list, type "<null>" (without the quotes), and press Return or Enter.
To view the current value of a macro or function variable	Move the cursor to the procedure text of the variable name and wait. On *Macintosh*, the value appears to the right of the debugger buttons. On *Windows*, the value appears in a tooltip window.
To view the current value of an expression	Select the expression text with the cursor, position the cursor over the selection, and wait.
	(Expressions involving user-defined functions will not be evaluated unless V_debugDangerously is set to 1.)
To view global values in the current data folder	Choose "local and global variables" from the debugger pop-up menu.
To view type information about variables	Choose "show variable types" from the debugger pop-up menu.
To resize the columns in the variables list	Drag a divider in the list to the left or right.
To show or hide the Waves, Structs, and Expressions pane	Drag the divider on the right side of the Variables list left or right.