

Chapter  
**IV-8**

## Debugging

Debugging Procedures .....	184
Debugging With Print Statements.....	184
The Debugger .....	184
Setting Breakpoints.....	185
Debugging on Error.....	185
Macro Execute Error: The Debug Button .....	186
Stepping Through Your Code.....	186
The Stack and Variables Lists .....	187
The Variables List Columns .....	188
Variables Pop-Up Menu .....	188
Function Variables.....	188
Macro Variables .....	189
Wave, Structures, and Expressions Pane .....	191
Expressions.....	192
Waves in Current or Root Data Folder.....	192
WAVES and STRUCTs.....	194
The Procedure Pane.....	195
After You Find a Bug.....	196
Debugger Shortcuts .....	196

### 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

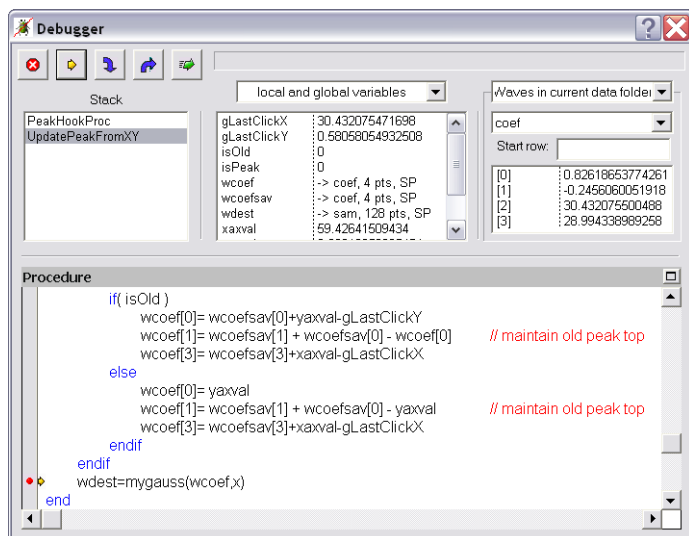
    <body of function>

    Print result
    return result
End
```

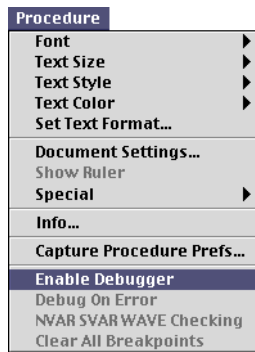
See **Creating Formatted Text** on page IV-230 for details on the `Printf` operation.

### The Debugger

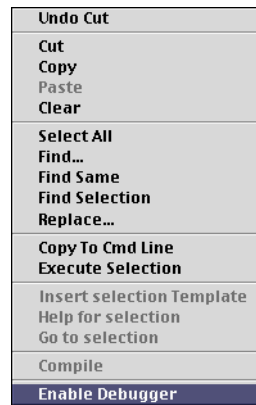
When a procedure doesn't produce the results you want, you can use Igor's built-in debugger to observe the execution of user-defined macros and functions while single-stepping through the lines of code.



The debugger is normally disabled. Enable it using either the Procedure menu or by Control-clicking (*Macintosh*) or right-clicking (*Windows*) in any procedure window to get the pop-up menu:



Procedure Menu



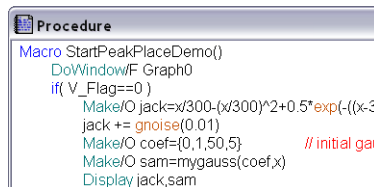
Contextual menu

The debugger window will automatically appear when one of the following events occurs:

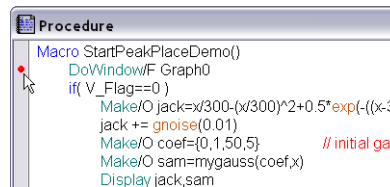
1. A breakpoint that you previously set is encountered.
2. An error occurs, and you have enabled debugging on that kind of error.
3. An error dialog is presented, and you click the Debug button.

## 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:



Debugger Disabled



Debugger Enabled

When a line of code marked with the red dot (denoting a set breakpoint) is about to execute, the debugger window will appear.

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.                          |

You can use the /Z flag to hide failures from SVAR, NVAR and WAVE checking. This is appropriate where the reference is subsequently checked with WaveExists, NVAR\_Exists, or SVAR\_Exists:

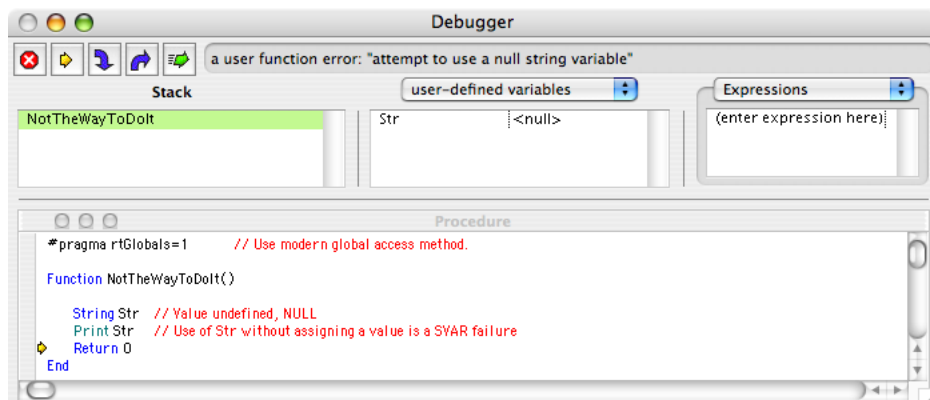
```
WAVE/Z wv=<pathToPossiblyMissingWave>

if( WaveExists(wv) )
  <do something with wv>
endif
```

See **Runtime Lookup of Globals** on page IV-50 for details.

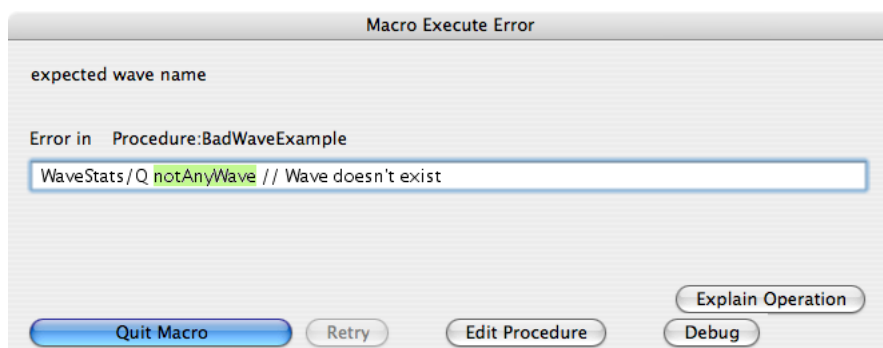
## Chapter IV-8 — Debugging

Use the Procedure or pop-up menus to choose either or both error categories. If the selected error occurs, the debugger will open and an error message will appear in the debugger window's status area. The error message was generated by the command *above* the yellow arrow:



### Macro Execute Error: The Debug Button

When the debugger is enabled and an error occurs in a Macro, an error dialog is presented that will (usually) have a Debug button in it. Click the button to open the debugger window.




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

When an error is reported by a function, a different dialog appears long after the error was actually committed. The Debug On Error option catches errors in functions immediately after they are committed.


### Stepping Through Your Code

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 ceases execution of the running function or macro before it completes. This is equivalent to clicking Igor's Abort button (*Windows*) or pressing Command-period (*Macintosh*) while the procedure is running.

Keyboard shortcuts: (none)

**Note:** 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 Over 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 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 executes until the current subroutine is exited, or an error or breakpoint is encountered.

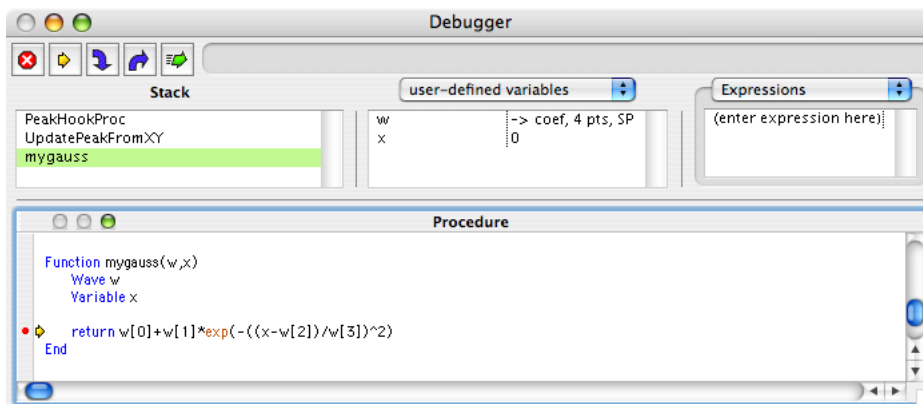
Keyboard shortcuts: -, \_ (underscore) or keypad -

 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

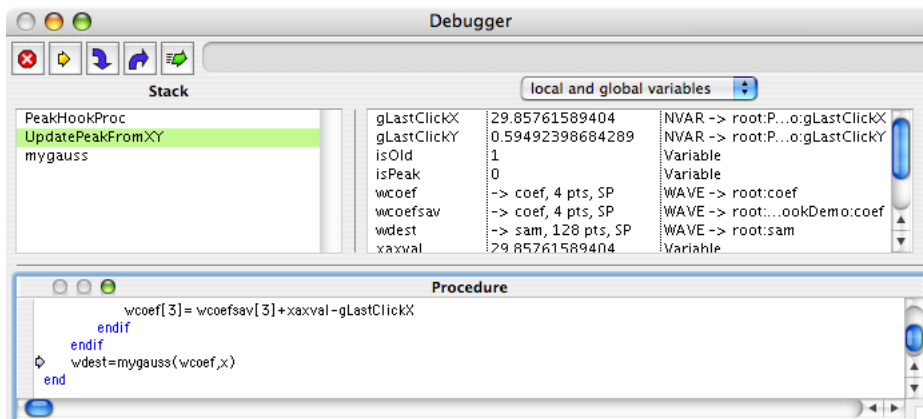
The Stack List shows the routine that is currently executing and the chain of routines that called it. The top item in the list is the routine that began execution and the bottom item is the routine which is currently executing.



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.

## Chapter IV-8 — Debugging

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.

The first column is the name of the local variable. Note that the name of an NVAR, SVAR, or WAVE reference is a local name referring to a global object.

The second column is the value of the local variable. Double-click the second column to edit strings or variables.

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.

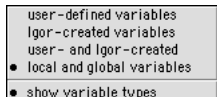
To determine the value of a particular wave element, use the **Wave, Structures, and Expressions Pane** on page IV-191.

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

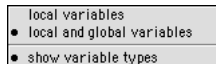
**Note:** The currentDF item is separated by a dashed line because currentDF is not really a global variable; it is a convenient name to identify the current data folder. See Chapter II-8, **Data Folders**, for more information about data folders.

### Variables Pop-Up Menu

The Variables pop-up menu controls which information is displayed in the Variables List. The menu items differ when the routine chosen from the Stack List is a function or a macro/proc:



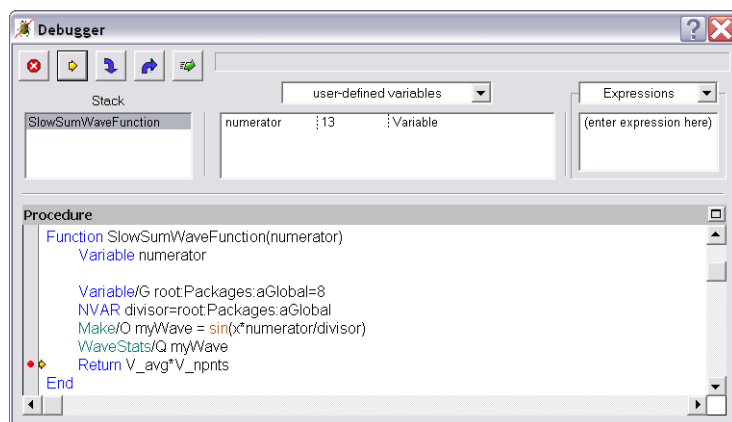
Pop-up Menu for Functions



Pop-up Menu for Macros,  
Procs, and Windows

### Function Variables

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

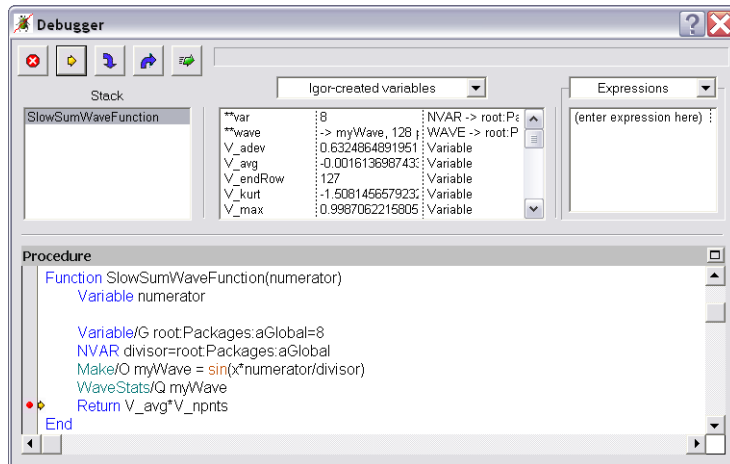


User-defined variables in functions include all items passed as parameters (numerator in this example) and any local strings and variables.

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.

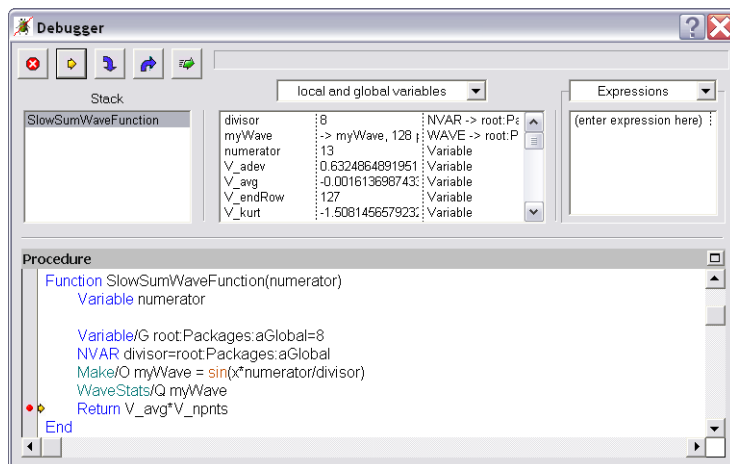
Use “Igor-created variables” to show local variables that Igor creates for functions when they call an operation or function that returns results in specially-named variables. The **WaveStats** operation (see page V-729), for example, defines V\_adev, V\_avg, and other variables to contain the statistics results:



**Note:** Some global references are created automatically for commands such as the Make operation; they have names that start with “\*\*”. These are shown only when Igor-created variables is selected.

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:

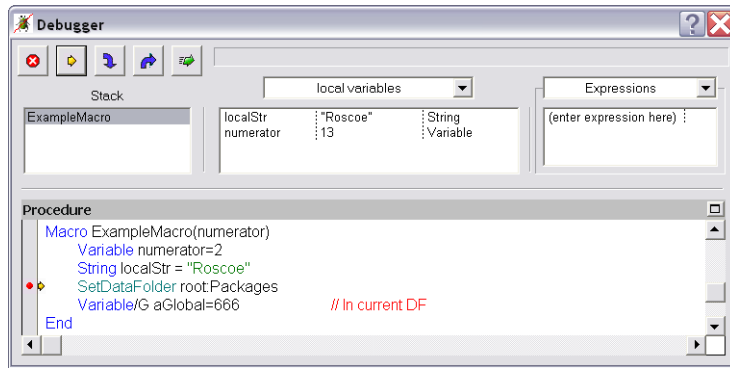


Choosing “local and global variables” also displays the current data folder at the end of the list as the mythical currentDF variable (see **The Variables List Columns** on page IV-188).

## Macro Variables

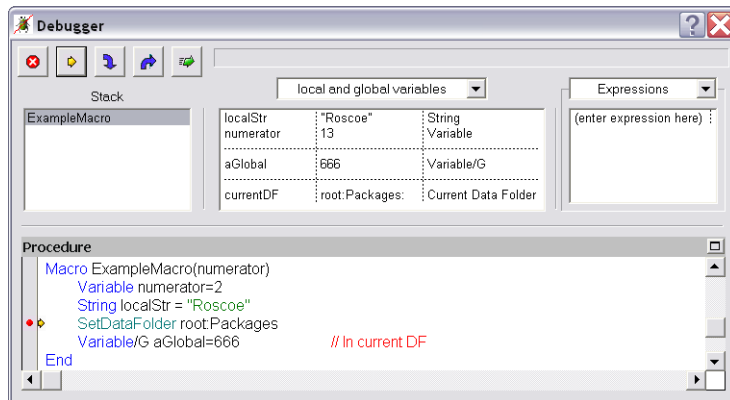
The ExampleMacro below illustrates how variables in Macros, Procs or Window procedures are classified as locals or globals:

## Chapter IV-8 — Debugging



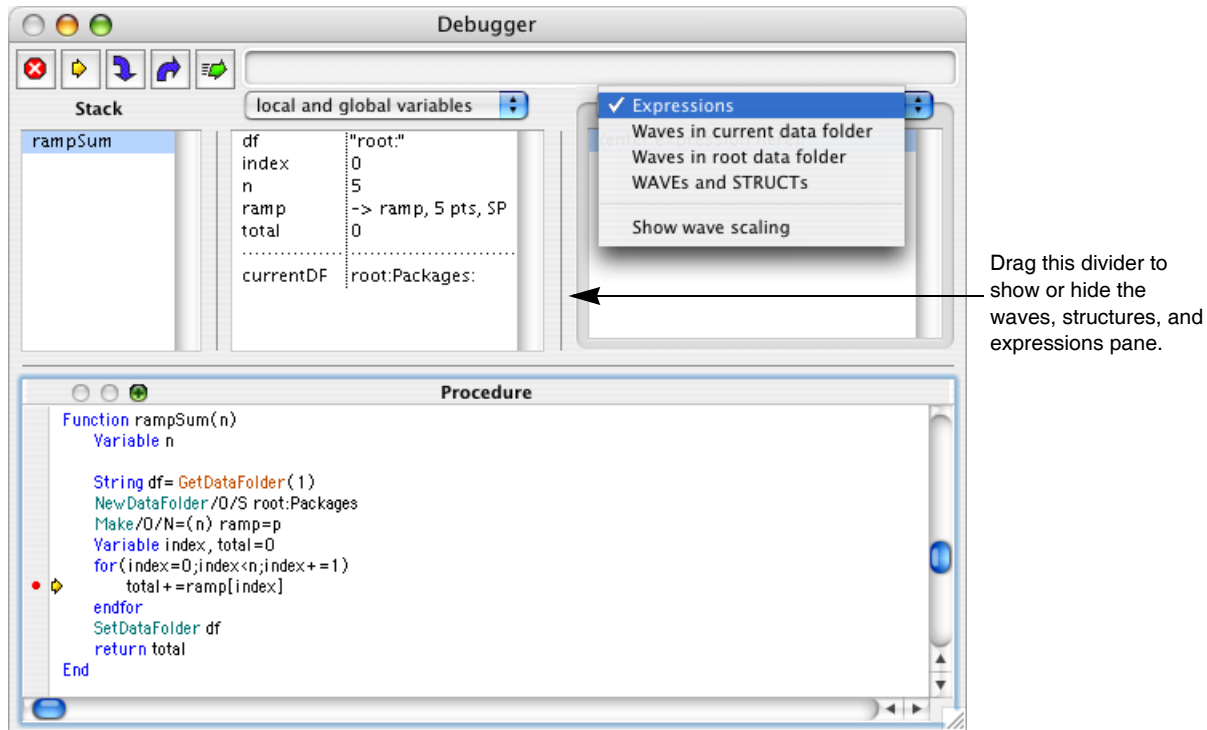
Local variables in macros include all items passed as parameters (numerator in this example) and local variables and local strings (localStr), and Igor-created local variables created by operations such as WaveStats.

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.



## Wave, Structures, and Expressions Pane

The waves, structures, and expressions pane is on the right side of the variables list:



This pane is hidden when there isn't enough room or 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.

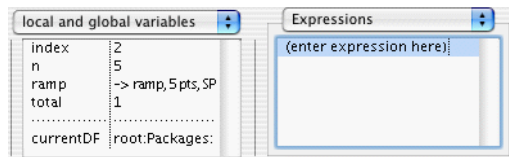
The pop-up menu controls what is shown in the pane:

Pop-up Menu Selection	Pane Contents
Expressions	Numeric or string expressions which are evaluated in the context of the selected function or macro.
Waves in current data folder	A list of waves in the current data folder and the contents of one selected wave.
Waves in root data folder	The same, but for the root: data folder.
WAVES and STRUCTs	A list of WAVE and STRUCT references in the selected function. Disabled when a macro or proc is selected in the Stack list.
Show wave scaling	When unchecked, wave indexes are shown using rows, columns, layers, and chunks. The values are enclosed in square brackets. When checked, wave indexes are shown using the scaled values (see <b>Waveform Model of Data</b> on page II-77) and the value are enclosed in parentheses

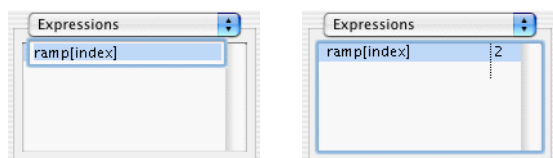
## Chapter IV-8 — Debugging

### Expressions

Replace the “(enter expression here)” prompt:



by clicking it, typing a numeric or string expression, and pressing Return.



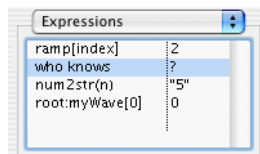
Adding an expression adds a blank row at the end of the list that can be double-clicked to enter another expression. You can edit any of the expressions by double-clicking and typing.

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

The result of the expression will be 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 “?”:

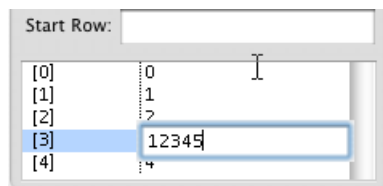


The expressions are discarded when a new Igor experiment is opened or when Igor quits.

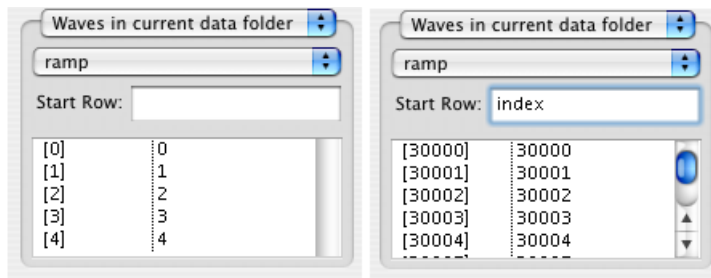
### Waves in Current or Root Data Folder

The debugger shows the waves in the specified data folder. You can select one of the waves to inspect a one-dimensional portion.

You can edit the value of a wave element by double-clicking the value column and editing the value there. Press return to enter the value.

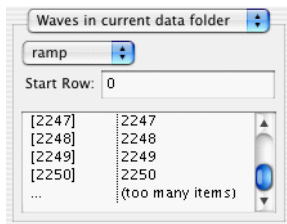


For a one-dimensional wave, you can specify the first row of the wave to display in the “Start Row:” control, or leave it blank to display starting with row 0. To look beyond the start of very long waves, you can enter a number or numeric expression into “Start Row:”



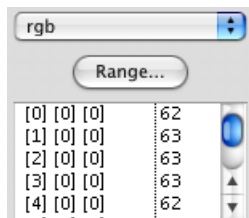
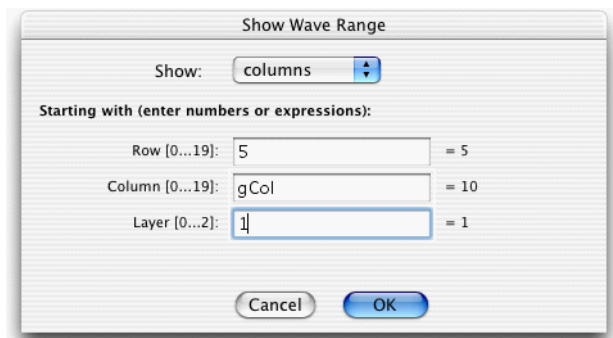
The wave value list will display only a limited number of rows, 1024 by default, in order to keep the debugger from being sluggish. When the list cannot display all of the rows, the last row in the list will say “(too many items)”. You can change the maximum number of rows displayed here with the SetIgorOption command:

`SetIgorOption DebuggerWaveRows=numRows`

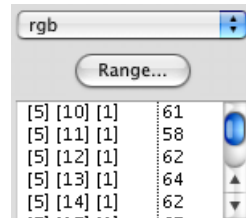


Use the Start Row to bring the row you wish to inspect within view.

Multidimensional waves can also be viewed, but only a one-dimensional subset. To view a particular range of values use the “Range...” button (which appears only when a multidimensional wave is chosen in the pop-up menu) and the resulting Show Wave Range dialog:



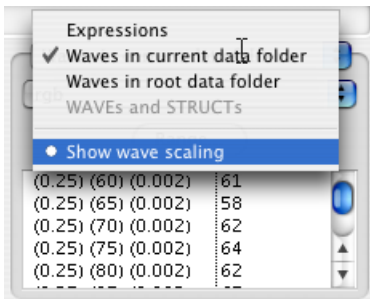
Default: rows 0 ... n of column 0, layer 0



rows 5 ... of column 10, layer 1

## Chapter IV-8 — Debugging

Choose "Show wave scaling" to display wave indexes using the scaled values (see **Waveform Model of Data** on page II-77). Scaled values are displayed in parentheses:



### WAVEs and STRUCTs

WAVE and STRUCT references are allowed only in functions, so this pop-up menu choice is disabled when the routine selected in the Stack list is a macro or proc.

A WAVE reference points to a global wave, which can be in any data folder.

A STRUCT reference points to a structure which exists on the runtime stack and does not exist as a global object in any data folder.

Each is composed of individual elements, and these can be inspected in the list in this pane.

Here's the code of an example contrived to demonstrate inspecting a structure's elements using the WAVEs and STRUCTs pane:

```
Structure stuff
  String traces
  Variable nTraces
  STRUCT traceStuff trace[2]
EndStructure

Structure traceStuff
  Variable ndx
  String trace
  WAVE w
EndStructure

Function top(graph)
  String graph

  STRUCT stuff s

  s.traces=TraceNameList(graph, ";", 1 )
  s.nTraces= ItemsInList(s.traces)

  STRUCT traceStuff ts

  initTrace(ts, 0, graph, s.traces)
  s.trace[0]= ts

  initTrace(ts,1, graph, s.traces)      // breakpoint set here
  s.trace[1]= ts
End

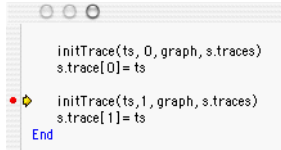
Function initTrace(ts, index, graph, traces)
  STRUCT traceStuff &ts
  Variable index
  String graph, traces
```

```

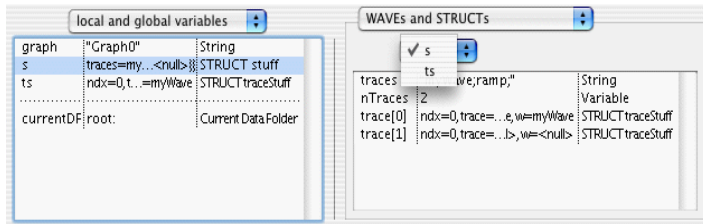
ts.ndx= index
ts.trace= StringFromList(index, traces)
WAVE ts.w=TraceNameToWaveRef(graph, ts.trace)
End

```

Running top("Graph0") with the breakpoint at the second initTrace call



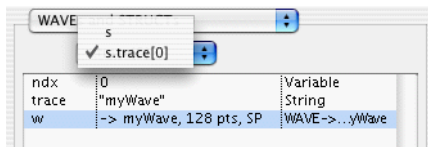
results in the Variables list shown below. You can see in the Variables list that "s" is the name of structure of type "stuff":



Double-clicking the structure s row in the Variables list (or selecting s from the pop-up menu of WAVES and STRUCTS) displays the content of the structure.

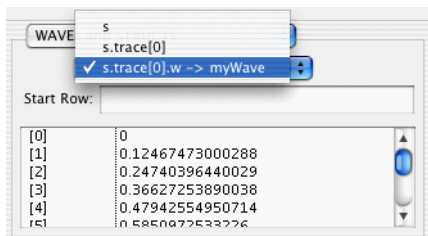
You can see that the listing of the contents of a structure on a single line is of limited use!

To see the contents of STRUCT traceStuff trace[0], double-click the trace[0] row in the WAVES and STRUCTS list:



You can edit the value of ndx and trace by double-clicking the second column.

To see the contents of s.trace[0].w, double-click w's row:



You can edit the values of myWave by double-clicking the second column.

Use the pop-up menu to view previous levels of the s structure. When you choose the top-level ("s"), the menu will again be filled with all the WAVES and STRUCTS in the currently selected function.

### 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.

## Chapter IV-8 — Debugging

---

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. On the Macintosh, the value is displayed in the top-right corner of the debugger. On Windows, a tooltip is displayed near the cursor.

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 ( <i>Macintosh</i> ) or right-clicking ( <i>Windows</i> ).
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 ( <i>Macintosh</i> ) or right-clicking ( <i>Windows</i> ).
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 ( <i>Macintosh</i> ) or right-clicking ( <i>Windows</i> ).
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 ( <i>Macintosh</i> ) or right-clicking ( <i>Windows</i> ).
To execute the next command	On <i>Macintosh</i> press Enter, keypad Enter, or Return. For <i>Windows</i> , 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.

---

Action	Shortcut
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 <i>Macintosh</i> , the value appears to the right of the debugger buttons. On <i>Windows</i> , 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.

---

