Two columns are assumed to be X followed by Y, and are graphed as Y versus X. More columns do not affect the graph, though they are shown in the table.

Example

```
// This hook function prints the first line of opened TEXT files
// into the history area
Function AfterFileOpenHook(refNum,file,pathName,type,creator,kind)
   Variable refNum,kind
   String file,pathName,type,creator
   // Check that the file is open (read only), and of correct type
   if( (refNum >= 0) && (CmpStr(type,"TEXT")==0) )  // also "text", etc.
      String line1
      FReadLine refNum, line1 // Read the line (and carriage return)
      Print line1       // Print line in the history area.
   endif
   return 0             // don't prevent MIME-TSV from displaying
End
```

See Also

BeforeFileOpenHook and **SetIgorHook**.

BeforeDebuggerOpensHook

BeforeDebuggerOpensHook(*errorInRoutineStr*, *stoppedByBreakpoint*)

BeforeDebuggerOpensHook is a user-defined function that Igor calls when the debugger window is about to be opened, whether by hitting a breakpoint or when Debug on Error is enabled.

BeforeDebuggerOpensHook can be used to prevent the debugger window opening for certain error codes or in selected user-defined functions when Debug on Error is enabled. This is a feature for advanced programmers only. Most programmers will not need it.

This hook does not work well for macros or procs, because their runtime errors don't automatically open the debugger, but instead present an error dialog from which the user manually enters the debugger by clicking the Debug button.

Parameters

errorInRoutineStr contains the name of the routine (function or macro) the debugger will be stopping in as a fully-qualified name, comprised of at least "`ModuleName#RoutineName`", suitable for use with FunctionInfo.

If the routine is in a regular module procedure window (see **Regular Modules** on page IV-236), *errorInRoutineStr* will be a triple name such as "`MyIM#MyModule#MyFunction`".

stoppedByBreakpoint is 0 if the debugger is about to be shown because of Debug on Error, or non-zero if the debugger encountered a user-set breakpoint (see **Setting Breakpoints** on page IV-213).

If a breakpoint exists at the line where an error caused the debugger to appear, *stoppedByBreakpoint* will be non-zero, even though the cause was Debug on Error.

Details

If BeforeDebuggerOpensHook returns 0 or NaN (or doesn't return a value), the debugger window is opened normally.

If it returns 1, the debugger window is not shown and program execution continues.

All other return values are reserved for future use.

Example

The following hypothetical example:

1. Prevents breakpoints from bringing up the debugger, unless DEBUGGING is defined.
2. Prints the name of the routine with the error, and the error message.
3. Beeps before the debugger appears.

```
Function ProvokeDebuggerInFunction()
   DebuggerOptions enable=1, debugOnError=1     // Enable debug on error

   ProvokeDebugger()
```