```
End

Function ProvokeDebugger()
   Variable var=0 // Put a breakpoint here.
                  // Without a #define DEBUGGING, the breakpoint is skipped.
   Make/O $""     // Cause an error
   Print "Back from bad Make command in function"
End

static Function BeforeDebuggerOpensHook(pathToErrorFunction,isUserBreakpoint)
   String pathToErrorFunction
   Variable isUserBreakpoint

   #ifndef DEBUGGING
      if( isUserBreakpoint )
         return 1 // Ignore user breakpoints we forgot to clear.
                  // Don't use this during development!
      endif
   #endif

   Print "stackCrawl = ", GetRTStackInfo(0)
   Print "FunctionInfo = ", FunctionInfo(pathToErrorFunction)

   // Don't clear errors unless you're preventing the debugger from appearing
   Variable clearErrors= 0
   Variable rtErr= GetRTError(clearErrors)   // Get the error #

   Variable substitutionOption= exists(pathToErrorFunction)== 3 ? 3 : 2
   String errorMessage= GetErrMessage(rtErr,substitutionOption)

   Beep     // Audible cue that the debugger is showing up!

   Print "Error \""+errorMessage+"\" in "+pathToErrorFunction+"

   return 0 // Return 0 to show the debugger; an unexpected error occurred.
End

•ProvokeDebuggerInFunction()  // Execute this in the command line
  stackCrawl =
      ProvokeDebuggerInFunction;ProvokeDebugger;BeforeDebuggerOpensHook;
  FunctionInfo =
      NAME:ProvokeDebugger;PROCWIN:Procedure;MODULE:;INDEPENDENTMODULE:;...
  Error "Expected name" in ProcGlobal#ProvokeDebugger
  Back from bad Make command in function
```

**See Also**

**SetWindow**, **SetIgorHook**, and **User-Defined Hook Functions** on page IV-280

**Static Functions** on page IV-105, **Regular Modules** on page IV-236, **Independent Modules** on page IV-238

**FunctionInfo**, **GetRTStackInfo**, **GetRTError**, **GetRTErrMessage**

**Conditional Compilation** on page IV-108

## AfterMDIFrameSizedHook

**`AfterMDIFrameSizedHook(`*`param`*`)`**

AfterMDIFrameSizedHook is a user-defined function that Igor calls when the Windows-only "MDI frame" (main application window) has been resized.

AfterMDIFrameSizedHook can be used to resize windows to fit the new frame size. See **GetWindow** kwFrame and **MoveWindow**.