You can use the same syntax to display a static function in a non-independent module procedure file using a module name instead of (or in addition to) the independent module name.s

procWinTitle can also be just an independent module name in brackets to display all procedure windows that belong to the named independent module and define the specified function:

```
DisplayProcedure/W=$"[myIM]" "HVAxisList"
```

Examples

```
DisplayProcedure "Graph0"
DisplayProcedure/B=Panel0 "MyOwnUserFunction"
DisplayProcedure/W=Procedure          // Shows the main Procedure window
DisplayProcedure/W=Procedure/L=5      // Shows line 5 (the sixth line)
DisplayProcedure/W=$"Wave Lists.ipf"
DisplayProcedure "moduleName#myStaticFunctionName"
SetIgorOption IndependentModuleDev=1
DisplayProcedure "WMGP#GizmoBoxAxes#DrawAxis"
```

See Also

Independent Modules on page IV-238

HideProcedures, **DoWindow**

ProcedureText, **ProcedureVersion**, **ModifyProcedure**

MacroList, **FunctionList**

do-while

```
do
   <loop body>
while(<expression>)
```

A do-while loop executes *loop body* until *expression* evaluates as FALSE (zero) or until a break statement is executed.

See Also

Do-While Loop on page IV-45 and **break** for more usage details.

DoAlert

DoAlert [/T=*titleStr*] *alertType*, *promptStr*

The DoAlert operation displays an alert dialog and waits for user to click button.

Parameters

alertType=*t*	Controls the type of alert dialog: *t*=0: Dialog with an OK button. *t*=1: Dialog with Yes button and No buttons. *t*=2: Dialog with Yes, No, and Cancel buttons.
promptStr	Specifies the text that is displayed in the alert dialog.

Flags

/T=*titleStr*	Changes the title of the dialog window from the default title.

Details

DoAlert sets the variable V_flag as follows:

1:	Yes clicked.
2:	No clicked.
3:	Cancel clicked.