Independent Modules - A Simple Example

Here is a simple example using an independent module. This code must be in its own procedure file and not in the built-in procedure file:

```
#pragma IndependentModule = IndependentModuleA

static Function Test()              // static means private to file
   Print "Test in IndependentModuleA"
End

// This must be non-static to call from command line (ProcGlobal context)
Function CallTestInIndependentModuleA()
   Test()
End
```

From the command line (the ProcGlobal context):

```
CallTestInIndependentModuleA()                     // Error

IndependentModuleA#CallTestInIndependentModuleA()  // OK

IndependentModuleA#Test()                          // Error
```

The first command does not work because the functions in the independent module are accessible only using a qualified name. The second command does work because it uses a qualified name and because the function is public (non-static). The third command does not work because the function is private (static) and therefore is accessible only from the file in which it is defined. A static function in an independent module is not accessible from outside the procedure file in which it is defined unless it is in an enclosed regular module as described under **Regular Modules Within Independent Modules** on page IV-242.

SetIgorOption IndependentModuleDev=1

By default, the debugger is disabled for independent modules. It can be enabled using:

```
SetIgorOption IndependentModuleDev=1
```

Also by default, independent module procedure windows are not listed in the Windows→Procedure Windows submenu unless you use `SetIgorOption IndependentModuleDev=1`.

When `SetIgorOption IndependentModuleDev=1` is in effect, the Windows→Procedure Windows submenu shows all procedure windows, and those that belong to an independent module are listed with the independent module name in brackets. For example:

```
DemoLoader.ipf [WMDemoLoader]
```

This bracket syntax is used in the **WinList**, **FunctionList**, **DisplayProcedure**, and **ProcedureText** functions and operations.

To get the user experience, as opposed to the programmer experience, return to normal operation by executing:

```
SetIgorOption IndependentModuleDev=0
```

Independent Module Development Tips

Development of an independent module may be easier if it is first done as for normal code. Add the module declaration

```
#pragma IndependentModule = moduleName
```

only after the code has been fully debugged and is working properly.