```
   Button button0,proc=RegularModuleA#ButtonProc
End
```

RegularModuleA is the name we have chosen for the regular module for demonstration purposes. You should choose a more descriptive module name.

The use of a qualified name, RegularModuleA#ButtonProc, allows Igor to find and execute the static ButtonProc function in the RegularModuleA module even though ButtonProc is running in the ProcGlobal context.

To protect the CreatePanel function from name conflicts we also made it static. To create the panel, execute:

```
RegularModuleA#CreatePanel()
```

Regular Modules and User-Defined Menus

Menu item execution text also runs in the ProcGlobal context. If you want to call a routine in a regular module you must use a qualified name.

Continuing the example from the preceding section, here is how you would write a menu definition:

```
#pragma ModuleName = RegularModuleA

Menu "Macros"
   "Create Panel", RegularModuleA#CreatePanel()
End
```

See also **Procedure Modules** on page IV-236, **Independent Modules** below, **Controls and Control Panels** on page III-413, **User-Defined Hook Functions** on page IV-280 and **User-Defined Menus** on page IV-125.

Independent Modules

An independent module is a set of procedure files that are compiled separately from all other procedures. Because it is compiled separately, an independent module can run when other procedures are in an uncompiled state because the user is editing them or because an error occurred in the last compile. This allows the independent module's control panels and menus to continue to work regardless of user programming errors.

Creating an independent module adds complexity and requires a solid understanding of Igor programming. You should use an independent module if it is important that your procedures be runnable at all times. For example, if you have created a data acquisition package that must run regardless of what the user is doing, that would be a good candidate for an independent module.

A file is designated as being part of an independent module using the IndependentModule pragma:

```
#pragma IndependentModule = imName
```

Make sure to use a distinctive name for your independent module.

The IndependentModule pragma is not allowed in the built-in procedure window which is always in the ProcGlobal module.

It is normal for multiple procedure files that are part of the same package to be in the same independent module.

An independent module creates an independent namespace. Function names in an independent module do not conflict with the same names used in other modules. To call an independent module function from the ProcGlobal module or from a regular module the function must be public (non-static) and you must use a qualified name as illustrated in the next section.

An independent module can call only procedures in that independent module.