dependencies to evaluate correctly, the dependency must use GetIndependentModuleName to create a formula to pass to the **SetFormula** operation. For example, outside of an independent module, this works:

```
String formula = "foo(root:wave0)"
SetFormula root:aVariable $formula
```

But inside an independent module you need this:

```
#pragma IndependentModule=MyIM
String formula = GetIndependentModuleName() + "#foo(root:wave0)"
SetFormula root:aVariable $formula
```

Independent Modules and Pictures

To allow DrawPICT to use a picture in the picture gallery, you must prepend GalleryGlobal# to the picture name:

```
DrawPICT 0,0,1,1,GalleryGlobal#PICT_0
```

Without GalleryGlobal, only proc pictures can be used in an independent module.

Making Regular Procedures Independent-Module-Compatible

You may want to make an existing set of procedures into an independent module. Alternatively, you may want to make an existing procedure independent-module-compatible so that it can be #included into an independent module. This section outlines the necessary steps.

1. If you are creating an independent module, add the IndependentModule pragma:
   ```
   #pragma IndependentModule=<NameOfIndependentModule>
   ```
2. Change any Macro or Proc procedures to functions.
3. Make Execute commands suitable for running in the ProcGlobal context or in an independent module using GetIndependentModuleName. See **Using Execute Within an Independent Module** on page IV-243.
4. Make PopupMenu controls that call a string function to populate the menu work in the ProcGlobal context or in an independent module using GetIndependentModuleName. See **Independent Modules and Popup Menus** on page IV-241.
5. Make any dependencies work in the ProcGlobal context or in an independent module using GetIndependentModuleName. See **Independent Modules and Dependencies** on page IV-243.

See also **Procedure Modules** on page IV-236, **Regular Modules** on page IV-236, **Controls and Control Panels** on page III-413, **User-Defined Hook Functions** on page IV-280, **User-Defined Menus** on page IV-125, and **GetIndependentModuleName** on page V-303.

Public and Static Functions

The static keyword marks a user-defined function as private to the file in which it is defined. It can be called within that file only, with an exception explained below for "regular modules".

If you are a beginning Igor programmer and you are not familiar with the advanced concepts of Regular Modules and Independent Modules, it is sufficient to think of this like this:

- A static function can be called only from the procedure file in which it is defined.
- A public function can be called from any procedure file.

The rest of this section is for advanced programmers with an understanding of **Regular Modules** and **Independent Modules**.

- A public function defined in the default ProcGlobal module can be called from another file in the ProcGlobal module or from a regular module without using a qualified name.
- A public function defined in a regular module can be called from a procedure file in ProcGlobal or