Wave dependencies are also deleted by operations that overwrite the values of their wave parameters. Some of these operations are:

```
FFT      Convolve        Correlate   Smooth   GraphWaveEdit
Hanning  Differentiate   Integrate   UnWrap
```

Dependencies can also be deleted using the Object Status dialog.

Broken Dependent Objects

Igor compiles the text of a dependency formula to low-level code and stores both the original text and the low-level code with the dependent object. At various times, Igor may need to recompile the dependency formula text. At that time, a compilation error will occur if:

- The dependency formula contains an error
- An object used in the dependency formula has been deleted, renamed, or moved
- The dependency formula references a user-defined function that is missing
- The dependency formula references a user-defined function and procedures are not compiled

When this happens, the dependent object will no longer update but will retain its last value. We call such an object “broken”.

To inspect broken objects, invoke the Object Status dialog. Choose Broken Objects from the pop-up menu above the status area. If there are any broken objects, they will appear in the Current Object pop-up. Select an object from the Current Object pop-up to inspect it.

When Dependencies are Updated

Dependency updates take place at the same time that graphs are updated. This happens after each line in a macro is executed, or when DoUpdate is called from a macro or user function, or continuously if a macro or function is not running.

Dependency formulas used as input to the SetBackground and ValDisplay operations, and in some other contexts, can alternately be specified as a literal string of characters using the following syntax:

```
#"text_of_the_dependency_expression"
```

Note that what follows the # char must be a literal string — not a string expression.

This sets the dependency formula *without* compiling it or checking it for validity. If you need to set the dependency formula of an object to something that is not currently valid but will be in the future, then use this alternate method.

Programming with Dependencies

You cannot use := to create dependencies in user-defined functions. Instead you must use the **SetFormula** operation (see page V-847).

```
Function TestFunc()
   Variable/G varNum=-666
   Make wave0
   SetFormula wave0, "varNum"    // Equivalent to wave0 := varNum
End
```