SetFileFolderInfo returns information about the file or folder in the following variables:

V_flag	0:	File or folder was found.
	-1:	User cancelled the Open File dialog.
	>0:	An error occurred, such as the specified file or folder does not exist.
S_path		File system path of the selected file or folder.

Examples

Change the file creator code; no complaint if it doesn't exist:

```
SetFileFolderInfo/Z /CRE8="CWIE", "Macintosh HD:folder:afile.txt"
```

Set the file modification date:

```
Variable mDate= Date2Secs(2000,12,25) + hrs*3600+mins*60+secs
SetFileFolderInfo/P=myPath/MDAT=(mDate), "afile.txt"
```

Remove read-only property from a folder and everything within it:

```
SetFileFolderInfo/P=myPath/D/R/RO=0
```

See Also

The **GetFileFolderInfo**, **MoveFile**, and **FStatus** operations. The **IndexedFile**, **date2secs**, and **ParseFilePath** functions.

SetFormula

SetFormula *waveName*, *expressionStr*

SetFormula *variableName*, *expressionStr*

The SetFormula operation binds the named wave, numeric or string variable to the expression or, if the expression is "", unbinds it from any previous expression. In user functions, SetFormula must be used to create dependencies.

Parameters

expressionStr is a string containing a numeric or string expression, depending on the type of the bound object.

Pass an empty string ("") for *expressionStr* to clear any previous dependency expression associated with the wave or variable.

Details

The dependent object (the wave or variable) will depend on the objects referenced in the string expression. The expression will be reevaluated any time an object referred to in the expression is modified.

Besides being set from a string expression this differs from just typing:

```
name := expression
```

in that syntax errors in *expressionStr* are not reported and are not fatal. You end up with a dependency assignment that is marked as needing to be recompiled. The recompilation will be attempted every time an object is created or when the procedure window is recompiled.

Use the Object Status dialog in the Misc menu to check up on dependent objects.

Examples

This command makes the variable v_sally dependent on the user-defined function anotherFunction, waves wave_fred and wave_sue, and the system variable K2:

```
SetFormula v_sally, "anotherFunction(wave_fred[1]) + wave_sue[0] + K2"
```

This is equivalent to:

```
v_sally := anotherFunction(wave_fred[1]) + wave_sue[0] + K2
```

except that no error will be generated for the SetFormula if, for instance, wave_fred does not exist.

A string variable dependency can be created by a command such as:

```
SetFormula myStringVar, "note(wave_joe)"
```

observe that *expressionStr* is a string *containing* a string expression, and that:

```
SetFormula myStringVar,note(wave_joe)
```