The **ControlUpdate**, **WaveList**, and **TraceNameList** operations.

**Special Characters in Menu Item Strings** on page IV-133.

# PopupMenuControl

**PopupMenuControl**

PopupMenuControl is a procedure subtype keyword that identifies a macro or function as being an action procedure for a user-defined pop-up menu control. See **Procedure Subtypes** on page IV-204 for details. See **PopupMenu** for details on creating a popup menu control.

# PossiblyQuoteName

**PossiblyQuoteName(*nameStr*)**

The PossiblyQuoteName function returns the input name string if it conforms to the rules of standard wave or Data Folder names. If it does not, then the name is returned in single quotes. This is used when generating a command string that you will pass to the Execute command. You might get the input name string from a function such as **NameOfWave** or **CsrXWave**.

**Examples**

```
Print PossiblyQuoteName("wave0")        // prints wave0
Print PossiblyQuoteName("wave 0")       // prints 'wave 0'
```

**Details**

See **Programming with Liberal Names** on page IV-168 for an example.

# Preferences

**Preferences** [**/Q**] [***newPrefsState***]

The Preferences operation sets or displays the state of user preferences.

User preferences affect the creation of *new* graphs, panels, tables, layouts, notebooks, procedure windows, and the command window. They also affect the appearance of waves appended to graphs and tables, and objects appended to layouts.

**Parameters**

If *newPrefsState* is present, it sets the state of user preferences as follows:

*newPrefsState*=0: Preferences off (use factory defaults).

*newPrefsState*=1: Preferences on.

If *newPrefsState* is omitted, the state of user preferences is printed in the history area.

**Flags**

/Q Disables printing to the history.

**Details**

The Preferences operation sets the variable V_flag to the state of user preferences that were in effect *before* the Preferences command executed: 1 for on, 0 for off.

You can also set the state of Preferences with the Misc menu.

Under most circumstances we want procedures to be independent of preferences so that a particular procedure will do the same thing regardless of the state of preferences. To achieve this, preferences are automatically off when you initiate procedure execution. When execution is complete, the state of preferences is restored to what it was before.

If you want preferences to be in effect during procedure execution, you must turn it on with the Preferences operation.

If the preferences setting is changed by a procedure, the effect of the call is propagated down the calling chain. If a macro changes the preferences setting, that change is undone when the macro returns. If a function changes the preferences setting, the change persists after the function returns. However, even with a function, the changed preferences state does not persist when Igor regains control.