WaveExists, **NVAR_Exists**, and **SVAR_Exists** to check the existence of expected objects and fail gracefully if they do not exist.

Data Folder Applications

There are two main uses for data folders:

- To store globals used by procedures to keep track of their state
- To store runs of data with identical structures

The following sections explore these applications.

Storing Procedure Globals

If you create packages of procedures, for example data acquisition, data analysis, or a specialized plot type, you typically need to store global variables, strings and waves to maintain the state of your package. You should keep all such items in a data folder that you create with a unique name to reduce clutter and avoid conflicts with other packages.

If your package is inherently global in nature, data acquisition for example, create your data folder within a data folder named Packages that you create in the root data folder. If your package is named MyDataAcq, you would store your globals in root:Packages:MyDataAcq. See **Managing Package Data** on page IV-249 for details and examples.

On the other hand, if your package needs to create a group of variables and waves as a result of an operation on a single input wave, it makes sense to create your data folder in the one that holds the input wave (see the **GetWavesDataFolder** function on page V-317). Similarly, if you create a package that takes an entire data folder as input (via a DFREF parameter pointing to the data folder) and needs to create a data folder containing output, it should create the output data folder in the one, or possibly at the same level as the one containing the input.

If your package creates and maintains windows, it should create a master data folder in root:Packages and then create individual data folders within the master that correspond to each instance of a window. For example, a Smith Chart package would create a master data folder as root:Packages:SmithCharts: and then create individual data folders inside the master with names that correspond to the individual graphs.

A package designed to operate on traces within graphs would go one step further and create a data folder for each trace that it has worked on, inside the data folder for the graph. This technique is illustrated by the Smooth Control Panel procedure file. For a demonstration, choose File→Examples→Feature Demos→Smoothing Control Panel.

Storing Runs of Data

If you acquire data using a well-established experimental protocol, your data will have a well-defined structure. Each time you run your protocol, you produce a new data set with the same structure. Storing each data set in its own data folder avoids name conflicts between corresponding items of different data sets. It also simplifies the writing of procedures to analyze and compare data set.

To use a trivial example, your data might have a structure like this:

```
<Run of data>
   String: conditions
   Variable: temperature
   Wave: appliedVoltage
   Wave: luminosity
```

Having defined this structure, you could then write procedures to:

1. Load your data into a data folder
2. Create a graph showing the data from one run
3. Create a graph comparing the data from many runs