```
   w = 123

   SetDataFolder root:
End

// String liberal NAME must NOT be quoted
Function DemoStringLiberalNames()
   SetDataFolder root:

   String dfName = "My Data Folder"       // String name must NOT be quoted
   NewDataFolder/O $dfName

   String wName = "My Wave"               // String name must NOT be quoted
   Make/O root:$(dfName):$wName

   Wave w = root:$(dfName):$wName         // String name must NOT be quoted
   w = 123

   SetDataFolder root:
End
```

The last example illustrates another subtlety. This command would generate an error at compile time:

```
Make/O root:$dfName:$wName       // ERROR
```

because Igor would intepret it as:

```
Make/O root:$(dfName:$wName)     // ERROR
```

To avoid this, you must use parentheses like this:

```
Make/O root:$(dfName):$wName     // OK
```

Runtime Lookup Example

In this example, a function named Routine calls another function named Subroutine and needs to access a number of result values created by Subroutine. To make it easy to clean up the temporary result globals, Subroutine creates them in a new data folder. Routine uses the results created by Subroutine and then deletes the temporary data folder.

```
Function Subroutine(w)
   WAVE w

   NewDataFolder/O/S SubroutineResults    // Results go here

   WaveStats/Q w              // WaveStats creates local variables
   Variable/G gAvg = V_avg    // Return the V_avg result in global gAvg
   Variable/G gMin = V_min
   String/G gWName = NameOfWave(w)

   SetDataFolder ::           // Back to original data folder
End

Function Routine()
   Make aWave= {1,2,3,4}
   Subroutine(aWave)

   DFREF dfr = :SubroutineResults

   NVAR theAvg = dfr:gAvg                    // theAvg is local name
   NVAR theMin = dfr:gMin
   SVAR theName = dfr:gWName
   Print theAvg, theMin, theName
```