Loading and Graphing Waveform Data

Here is a very simple example designed to show the basic form of an Igor function for automatically loading and graphing the contents of a data file. It loads a delimited text file containing waveform data and then makes a graph of the waves.

This example uses an Igor symbolic path. If you are not familiar with the concept, see **Symbolic Paths** on page II-22.

In this function, we make the assumption that the files that we are loading contain three columns of waveform data. Tailoring the function for a specific type of data file allows us to keep it very simple.

```
Function LoadAndGraph(fileName, pathName)
   String fileName      // Name of file to load or "" to get dialog
   String pathName      // Name of path or "" to get dialog

   // Load the waves and set the local variables.
   LoadWave/J/D/O/P=$pathName fileName
   if (V_flag==0)       // No waves loaded. Perhaps user canceled.
      return -1
   endif

   // Put the names of the three waves into string variables
   String s0, s1, s2
   s0 = StringFromList(0, S_waveNames)
   s1 = StringFromList(1, S_waveNames)
   s2 = StringFromList(2, S_waveNames)

   Wave w0 = $s0                // Create wave references.
   Wave w1 = $s1
   Wave w2 = $s2

   // Set waves' X scaling, X units and data units
   SetScale/P x, 0, 1, "s", w0, w1, w2
   SetScale d 0, 0, "V", w0, w1, w2

   Display w0, w1, w2           // Create a new graph

   // Annotate graph
   Textbox/N=TBFileName/A=LT "Waves loaded from " + S_fileName

   return 0                     // Signifies success.
End
```

s0, s1 and s2 are local string variables into which we place the names of the loaded waves. We then use the $ operator to create a reference to each wave, which we can use in subsequent commands.

Once the function is entered in the procedure window, you can execute it from the command line or call it from another function. If you execute

```
   LoadAndGraph("", "")
```

the LoadWave operation displays an Open File dialog allowing you to choose a file. If you call LoadAndGraph with the appropriate parameters, LoadWave loads the file without presenting a dialog.

You can add a “Load And Graph” menu item by putting the following menu declaration in the procedure window:

```
Menu "Macros"
   "Load And Graph...", LoadAndGraph("", "")
End
```

Because we have not used the “Auto name & go” option for the LoadWave operation, LoadWave displays another dialog in which you can enter names for the new waves. If you want the procedure to be more auto-

matic, use /A or /N to turn "Auto name & go" on. If you want the procedure to specify the names of the loaded waves, use the /B flag. See the description of the **LoadWave** operation (see page V-508) for details.

To keep the function simple, we have hard-coded the X scaling, X units and data units for the new waves. You would need to change the parameters to the SetScale operation to suit your data. For more flexibility, you would add additional parameters to the function.

It is possible to write LoadAndGraph so that it can handle files with any number of columns. This makes the function more complex but more general.

For more advanced programmers, here is the more general version of LoadAndGraph.

```
Function LoadAndGraph(fileName, pathName)
   String fileName      // Name of file to load or "" to get dialog
   String pathName      // Name of path or "" to get dialog

   // Load the waves and set the variables.
   LoadWave/J/D/O/P=$pathName fileName
   if (V_flag==0)       // No waves loaded. Perhaps user canceled.
      return -1
   endif

   Display              // Create a new graph

   String theWave
   Variable index=0
   do                   // Now append waves to graph
      theWave = StringFromList(index, S_waveNames) // Next wave
      if (strlen(theWave) == 0)              // No more waves?
            break                            // Break out of loop
      endif
      Wave w = $theWave
      SetScale/P x, 0, 1, "s", w             // Set X scaling
      SetScale d 0, 0, "V", w                // Set data units
      AppendToGraph w
      index += 1
   while (1)         // Unconditionally loop back up to "do"

   // Annotate graph
   Textbox/A=LT "Waves loaded from " + S_fileName

   return 0          // Signifies success.
End
```

The do-loop picks each successive name out of the list of names in S_waveNames and adds the corresponding wave to the graph. S_waveNames will contain one name for each column loaded from the file.

There is one serious shortcoming to the LoadAndGraph function. It creates a very plain, default graph. There are four approaches to overcoming this problem:

- Use preferences
- Use a style macro
- Set the graph formatting directly in the procedure
- Overwrite data in an existing graph

Normally, Igor does not use preferences when a procedure is executing. To get preferences to take effect during the LoadAndGraph function, you would need to put the statement "Preferences 1" near the beginning of the function. This turns preferences on just for the duration of the function. This will cause the Display and AppendToGraph operations to use your graph preferences.

Using preferences in a function means that the output of the function will change if you change your preferences. It also means that if you give your function to a colleague, it will produce different results. This