Because we loaded 'Raster Image #0' as a plain dataset, not as a formal image, the image plot is gray instead of colored. Also, it is rotated 90 degrees relative to the image plot we saw earlier in the tour. That's because Igor plots 2D data different from most programs and the HDF5LoadData operation does not compensate by default.

Our procedure is of rather limited value because it is hard-coded to use a specific symbolic path and a specific file name. We will now make it more general.

Before we do that, we will save the current work environment so that, if you make a mistake, you can revert to a version of it that worked.

8. **Choose File→Save Experiment As and save the current work environment in a new Igor experiment file named "HDF5 Tour.pxp" in a directory where you store your own files.**

   This saves all of your data and procedures in a single file. If need be, later you can revert to the saved state by choosing File→Revert Experiment.

9. **Open the Procedure window (Windows→Procedure Window) and replace the first two lines of the TestLoadDataset function with this:**

```
Function TestLoadDataset(pathName, fileName, datasetName)
    String pathName         // Name of symbolic path
    String fileName         // Name of HDF5 file
    String datasetName      // Name of dataset to be loaded
```

10. **Change the HDF5OpenFile command to this:**

```
HDF5OpenFile /P=$pathName /R /Z fileID as fileName
```

   Here we replaced HDF5Samples with $pathName. $pathName tells Igor to use the contents of the string parameter pathName as the parameter for the /P flag. We also replaced "TOVSB1NF.h5" with fileName so that we can specify the file to be loaded when we call the function instead of when we code it.

11. **Click the Compile button and then execute this in the command line:**

```
TestLoadDataset("HDF5Samples", "TOVSB1NF.h5", "Raster Image #0")
```

   This does the same thing as the earlier version of the function but now we have a more general function that can be used on any file in any directory.

   The command above reloaded the 'Raster Image #0' dataset into Igor, overwriting the previous contents. Since the new contents is identical to the previous contents, the graph and table did not change.

12. **Choose File→Save Experiment to save your current work environment in the HDF5 Tour.pxp experiment file that you created earlier.**

   You can now take a break and quit Igor if you want.

Saving HDF5 Data Programmatically

Now that we have seen how to programmatically load HDF5 data we will turn our attention to saving Igor data in an HDF5 file.

In this section, we will create some Igor data and save it in an HDF5 dataset from a user-defined function.

1. **If the HDF5 Tour.pxp experiment that you previously saved is not already open, open it by choosing File→Recent Experiments→HDF5 Tour.pxp.**

   Next we will create some data that we can save in an HDF5 file. We will create the data in a new Igor data folder to keep it separate from our other data.

2. **Choose Data→Data Browser. Click the New Data Folder button. Enter the name Test Data, click the Set As Current Data Folder and click OK.**

   The Data Browser shows the new data folder and a red arrow pointing to it. The red arrow indicates the current data folder. Operations that do not explicitly address a specific data folder work in the current data folder.

3. **On Igor's command line, execute these commands:**

```
Make/N=100 data0, data1, data2
```

```
SetScale x 0, 2*PI, data0, data1, data2
data0=sin(x); data1=2*sin(x+PI/6); data2=3*sin(x+PI/4)
Display data0, data1, data2
```

The SetScale operation set the X scaling for each wave to run from 0 to 2π. The symbol x in the wave assignment statements takes on the X value for each point in the destination wave as the assignment is executed.

(If you have not already done it, later you should do the Igor guided tour by choosing Help→Getting Started. It explains X scaling and wave assignment statements as well as many other Igor features.)

4. **Choose Misc→New Path and create an Igor symbolic path named HDF5Data pointing to a directory on your hard disk where you will save data.**

   Choose Misc→Path Status and verify that the HDF5Data symbolic path exists and points to the intended directory.

   Now that we have done some more work worth saving we will save the current experiment so we can revert to a known good state if necessary.

5. **Choose File→Save Experiment to save your current work environment in the HDF5 Tour.pxp experiment file that you created earlier.**

   If need be, later you can revert to the saved state by choosing File→Revert Experiment.

6. **Choose Windows→Procedure Window and paste the following into the Procedure window below the TestLoadDataset function:**

```
Function TestSaveDataset(pathName, fileName, w)
    String pathName         // Name of symbolic path
    String fileName         // Name of HDF5 file
    Wave w                  // The wave to be saved

    Variable result = 0     // 0 means no error

    Variable fileID

    // Create a new HDF5 file, overwriting if same-named file exists
    HDF5CreateFile/P=$pathName /O /Z fileID as fileName
    if (V_flag != 0)
        Print "HDF5CreateFile failed"
        return -1
    endif

    // Save wave as dataset
    HDF5SaveData /O /Z w, fileID
    if (V_flag != 0)
        Print "HDF5SaveData failed"
        result = -1
    endif

    // Close the HDF5 file
    HDF5CloseFile fileID

    return result
End
```

Read through the procedure. It should look familiar as it is similar to the TestLoadDataset function we wrote before. Again it has parameters that specify a symbolic path and file name. It does not have a parameter to specify the dataset name because HDF5SaveData uses the wave name as the dataset name unless instructed otherwise. This function has a wave parameter through which we will specify the wave whose data is to be written to the HDF5 file.

Here we use HDF5CreateFile to create a new file rather than HDF5OpenFile. HDF5CreateFile creates a new file and opens it, returning a file ID. If you wanted to add a dataset to an existing file you would use HDF5OpenFile instead of HDF5CreateFile.

HDF5CreateFile sets the local variable fileID to a value returned from the HDF5 library. We pass that value to the HDF5SaveData operation. The /O flag means that, if there is already a dataset with the same name, it will be overwritten.

Finally we call HDF5CloseFile to close the file that we opened via HDF5CreateFile.

7. **Click the Compile button at the bottom of the Procedure window.**

   If you get an error then you have not pasted the right text into the Procedure window or you have entered other incorrect text. If you can not find the error, choose File→Revert Experiment and go back to step 6 of this section.

   Now we are ready to run our procedure.

8. **Execute the following command in the Igor command line either by typing it and pressing Enter, by copy/paste followed by Enter, or by selecting it and pressing Control-Enter (Macintosh) or Ctrl-Enter (Windows):**

   ```
   TestSaveDataset("HDF5Data", "SaveTest.h5", data0)
   ```

   At this point the procedure should have correctly executed and you should see no error messages printed in the history area of the command window (just above the command line).

   Now we will verify that the data was saved.

9. **Choose Data→Load Waves→New HDF5 Browser. Click the Open HDF5 File button and open the SaveTest.h5 file that we just created.**

   Verify that the file contains a dataset named data0.

   The data0 dataset has some attributes. These attributes allow HDF5LoadData to fully recreate the wave and all of its properties if you ever load it back into Igor. If you don't want to save these attributes you can use the /IGOR=0 flag when calling HDF5SaveData.

10. **Click the Close HDF5 File button.**

    We can't write more data to the file while it is open in the HDF5 Browser.

    The TestSaveDataset function is rather limited because it saves just one wave. We will now make it more general so that it can save any number of waves. But first we will save our work since it is in a known good state.

11. **Choose File→Save Experiment to save your current work environment in the HDF5 Tour.pxp experiment file that you created earlier.**

    If need be, later you can revert to the saved state by choosing File→Revert Experiment.

    Next we will create a new, more general user function with a different name (TestSaveDatasets instead of TestSaveDataset).

12. **Choose Windows→Procedure Window and paste the following into the Procedure window below the TestSaveDataset function:**

```
Function TestSaveDatasets(pathName, fileName, listOfWaves)
    String pathName             // Name of symbolic path
    String fileName             // Name of HDF5 file
    String listOfWaves          // Semicolon-separated list of waves

    Variable result = 0         // 0 means no error

    Variable fileID

    // Create a new HDF5 file, overwriting if same-named file exists
    HDF5CreateFile/P=$pathName /O /Z fileID as fileName
    if (V_flag != 0)
       Print "HDF5CreateFile failed"
       return -1
    endif

    String listItem
    Variable index
```

```
    index = 0
    do
       listItem = StringFromList(index, listOfWaves)
       if (strlen(listItem) == 0)
          break // No more waves
       endif

       // Create a local reference to the wave
       Wave w = $listItem

       // Save wave as dataset
       HDF5SaveData /O /Z w, fileID
       if (V_flag != 0)
          Print "HDF5SaveData failed"
          result = -1
          break
       endif

       index += 1
    while(1)

    // Close the HDF5 file
    HDF5CloseFile fileID

    return result
End
```

Read through the procedure. It is similar to the TestSaveDataset function.

The first difference is that, instead of passing a wave, we pass a list of wave names in a string parameter. This parameter is a semicolon-separated list which is a commonly-used programming technique in Igor.

The next difference is that we have a do-while loop which extracts a name from the list and saves the corresponding wave as a dataset in the HDF5 file.

The statement

```
Wave w = $listItem
```

creates a local "wave reference" and allows us to use w to refer to a wave whose identity is determined by the contents of the listItem string variable. This also is a common Igor programming technique and is explained in detail in the Programming help file.

13. **Click the Compile button at the bottom of the Procedure window.**

    If you get an error then you have not pasted the right text into the Procedure window or you have entered other incorrect text. If you can not find the error, choose File→Revert Experiment and go back to step 12 of this section.

    Now we are ready to run our procedure.

14. **Enter the following command in the Igor command line either by typing it and pressing enter, by copy/paste followed by Enter, or by selecting it and pressing Control-Enter (Macintosh) or Ctrl-Enter (Windows):**

    ```
    TestSaveDatasets("HDF5Data", "SaveTest.h5", "data0;data1;data2;")
    ```

    The third parameter is a string in this function. In the previous example, it was a name, specifically the name of a wave. Strings are entered in double-quotes but names are not.

    At this point the procedure should have correctly executed and you should see no error messages printed in the history area of the command window (just above the command line).

    Now we will verify that the data was saved.

15. **Activate the HDF5 Browser window. Click the Open HDF5 File button and open the SaveTest.h5 file that we just created.**

    Verify that the file contains datasets named data0, data1 and data2.

16. **Click the Close HDF5 File button.**