binary when doing a conversion but it is up to you to mark such waves appropriately. Here is an example that illustrates the utility of marking a text wave as containing binary data:

```
// Mark contents as binary
SetWaveTextEncoding 255, 16, root:MyBinaryTextWave
// Convert contents to UTF-8 skipping binary waves
SetWaveTextEncoding /DF={root:,1} /CONV=1 1, 16
```

The first command marks a particular wave's content (16) as containing binary (255) as opposed to text. The second command converts all text wave text to UTF-8 (1) except that it automatically skips waves marked as binary.

This is fine but it would be tedious to identify and mark each text wave containing binary data. So SetWaveTextEncoding provides a way to do it automatically:

```
// Mark all binary waves as binary
SetWaveTextEncoding /BINA=1 /DF={root:,1} 255, 16
// Convert to UTF-8 skipping binary waves
SetWaveTextEncoding /DF={root:,1} /CONV=1 1, 16
```

The first command goes through each wave in the experiment. If it is a text wave, it examines its contents. If the wave contains control characters (codes less than 0x20) other than carriage-return (0x0D), linefeed (0x0A) and tab (0x09), it marks the wave as containing binary. Then the second command converts all non-binary text waves to UTF-8. These commands can be combined into one:

```
// Mark as binary then convert non-binary to UTF-8
SetWaveTextEncoding /BINA=1 /DF={root:,1} /CONV=1 1, 16
```

NOTE: The heuristic used by the /BINA flag is not foolproof. While most binary data will contain 0x00 bytes or other bytes that flag it as binary, it is possible to have binary data that contains no such bytes. In that case, the first command would fail to mark the text waves as containing binary and the second command would either return an error or clobber the binary data. For this reason, it is imperative that you back up any data before doing text encoding conversions.

PROGRAMMER'S NOTE: If you are a programmer and you create text waves to store binary data, be sure to mark them as binary. For example:

```
Wave/T bw = <path to some text wave used to contain binary data>
SetWaveTextEncoding 255, 16, bw        // Mark text wave as binary
```

Doing this will ensure that your binary data is not "converted" to some text encoding, thereby clobbering it.

Handling Text Waves Containing Binary Data

Usually you use a text wave containing binary data only as a container to be accessed by procedures that understand how to interpret the contents. You normally should not treat it as text. For example, you should not print its contents to the history because this treat the wave's binary contents as text. If you do treat it as text you are likely to get gibberish or a Unicode conversion error.

You can view a text wave marked as binary in a table and you can cut and paste its cells but you can not edit the binary data in the table's entry line. You can dump the contents of a cell as hex to the history area by pressing Cmd (*Macintosh*) or Ctrl (*Windows*) and double-clicking the cell.

Igor normally does automatic text encoding conversion when fetching data from a text wave or storing data into a text wave. For example:

```
// Make a wave to contain text in UTF-8 text encoding
Make /O /T UTF8 = {"<some Japanese text>"}      // UTF-8 by default

// Make a wave to contain text in Shift JIS text encoding
Make /O /T /N=1 ShiftJIS                       // UTF-8 by default
SetWaveTextEncoding 4, 16, ShiftJIS            // Now Shift JIS
```