- What Igor version you are running
- What incorrect output you are seeing and what you expect to see

LoadWave Text Encodings for Igor Binary Wave Files

The rules that LoadWave uses to determine the source text encoding of an Igor binary wave file are the same as the rules described under **Determining the Text Encoding for Wave Elements** on page III-473. The LoadWave /ENCG flag is ignored when loading an Igor binary wave file.

LoadWave Text Encodings for Plain Text Files

This section describes the rules that the LoadWave operation uses to determine the source text encoding when loading a plain text file. This includes general text, delimited text and Igor text files but not Igor binary wave files.

The rules that LoadWave uses to determine the source text encoding of a plain text file are the same as the rules described under **Determining the Text Encoding for a Plain Text File** on page III-467. The "specified text encoding", which is one of the factors used by the rules, is unknown unless you use the /ENCG flag to tell LoadWave the file's text encoding.

For further details see **LoadWave Text Encoding Issues** on page II-149.

Text Waves Containing Binary Data

You can set a text wave's content text encoding to the special value 255 using **SetWaveTextEncoding**. This marks a text wave as really containing binary data, not text.

Text is data that is represented by numeric codes which map to characters using some recognized text encoding such as MacRoman, Windows-1252, Shift JIS or UTF-8. By contrast, binary data does not follow any recognized text encoding and generally does not map to characters. Whereas text data is typically intended to be readable by humans using a text editor, binary data is intended to be processed only by programs.

You can mark any text wave element as binary but it is normally done only for text wave content because it is rare to store binary data in wave units, wave notes or wave dimension labels.

Igor text waves were designed to store text but are able to store any collection of bytes.

While a numeric wave stores a fixed number of bytes in each element, a text wave has the ability to store a different number of bytes in each point. For example:

```
Make /O /T /N=2 twave0
twave0[0] = "Hello"                 // 5 bytes
twave0[1] = "Goodbye"               // 7 bytes
```

This capability makes a text wave a convenient way to store a collection of binary arrays of different length. For example, you could store the contents of a different downloaded file in each element of a text wave:

```
Make /O /T /N=2 twave0
twave0[0] = FetchURL("http://www.wavemetrics.net/images/tbg.gif")
twave0[1] = FetchURL("http://www.wavemetrics.net/images/mbg.gif")
```

While most text waves contain text data, clever programmers sometimes use text waves to store binary data. Since binary data does not follow any recognized text encoding, it must not be treated as text. For example, you can convert a text wave from Shift JIS to UTF-8 and preserve the meaning of the text; you are merely changing how the characters are encoded. By contrast, if you attempt to convert binary data which you have mistaken for text, you turn your data into garbage.

The **SetWaveTextEncoding** operation can convert text waves from one text encoding to another but you never want to convert binary text waves. SetWaveTextEncoding conveniently skips text waves marked as