The most straight-forward character encoding scheme is UTF-32. The code unit is a 32-bit value and each code point is directly represented by a single code unit. UTF-32 is not widely used for historical reasons and because it is not memory-efficient.

UTF-16 is a widely-used character encoding scheme in which the code unit is a 16-bit value and each code point is represented by either one or two code units. Nearly all of the commonly-used characters require just one. Both Mac OS X and Windows use UTF-16 internally.

UTF-8 is a widely-used character encoding scheme in which the code unit is a byte and each code point is represented by one, two, three or four code units. The ASCII characters are represented by a single byte using the same code as in ASCII itself. All other characters are represented by two, three or four bytes. UTF-8 is the predominant character encoding scheme on the Worldwide Web and in email. It is the scheme used internally by Igor7 and later.

Problems Caused By Text Encodings

The use of different text encodings on different operating systems, for different languages, and in different versions of Igor requires that Igor convert between text encodings when reading files. In Igor7 and later this means converting from the file's text encoding to UTF-8 which Igor now uses for internal storage. Since ASCII text is the same in UTF-8 as in other byte-oriented text encodings, conversion and the problems it can cause are issues only for files that contain non-ASCII text.

Broadly speaking, when Igor opens an Igor6 file, three types of text encoding-related problems can occur:

- Igor's idea of the file's text encoding might be wrong (text encoding misidentification)
- The file may contain invalid byte sequences for its text encoding (invalid text)
- The file may contain a mixture of text encodings (mixed text encodings)

Text Encoding Misidentification Problems

This problem stems from the fact that it is often unclear what text encoding a given file uses. The clearest example of this is the plain text file - the format used for text data files, Igor procedure files and plain text notebooks. A plain text file usually stores just the text and nothing else. This means that there is often nothing in a plain text file that tells Igor what text encoding the file uses.

The problem is similar for plain text items stored within Igor experiment files. These include:

- wave names, wave units, wave notes, wave dimension labels
- text wave contents
- the experiment recreation procedures stored in an experiment file
- the history area contents
- the built-in procedure window's contents
- packed procedure files
- packed plain text notebook files

Igor Pro 6.30 and later, which we will call Igor6.3x for short, store text encoding information in the experiment file for these items but earlier versions of Igor did not.

The text encoding information stored by Igor6.3x and later is not foolproof. Igor6.3x determines the text encoding for a procedure file or a plain text notebook based on the font used to display it. Usually this is correct but it is not guaranteed as the Igor6 user can change a font at any time.

Waves created pre-Igor6.3x have unknown text encodings for their various items.

Igor6.3x determines the text encoding for wave items when the wave is created based on the text encoding used to enter text into the user interface (e.g., into dialogs). This will be the system text encoding except that in IgorJ (the Japanese version of Igor), it is Japanese regardless of the system text encoding. Usually this pro-