ConvertGlobalStringTextEncoding can convert all global string variables in the current experiment at once. Choose Misc→Text Encoding→Convert Global String Text Encoding to generate a ConvertGlobalString-TextEncoding command.

If you don't print or display a string variable or compare it to other text then its text encoding does not matter. For example, if you have a string variable that contains binary data and it is treated by your procedures as binary data, then everything will work out.

String Variable Text Encoding Error Example

This section gives a concrete example of a text encoding error which may help you to understand text encoding issues. As you read the example, keep these thoughts in mind:

- A text wave stores the raw bytes representing its text and an explicit setting that tells Igor the text encoding of the raw bytes. When you access an element of a text wave, Igor converts it to UTF-8 if it is not already stored as UTF-8 and is not marked as binary. During this conversion Igor uses the wave's explicit text encoding setting.
- String variables and other string expressions do not have text encoding settings and are always assumed to contain text encoded as UTF-8.

This example shows how you can create a text encoding problem and how you can avoid it.

```
Function TextEncodingErrorExample()
   // Make a text wave. Its text encoding defaults to UTF-8.
   Make /O/T/N=1 textWave0

   // Create a string variable containing non-ASCII text (the character ¹).
   // Igor converts literal text ("¹ != 3" in this case) to UTF-8
   // and stores it in the string variable.
   String text = "¹ != 3"                   // Stored as UTF-8

   // Convert the string variable contents to MacRoman.
   // Since the string has no text encoding setting, Igor still assumes it
   // contains UTF-8 but we know it really contains MacRoman.
   Variable textEncodingUTF8 = TextEncodingCode("UTF-8")
   Variable textEncodingMacRoman = TextEncodingCode("MacRoman")
   text = ConvertTextEncoding(text,textEncodingUTF8,textEncodingMacRoman,1,0)

   // Store the MacRoman text in the text wave. This creates bad data
   // because Igor assumes you are storing UTF-8 text in a UTF-8 text wave
   // when in fact you are storing MacRoman into a UTF-8 text wave.
   textWave0 = text                         // Store incorrect data

   // Print - an incorrect character will be printed because the wave's
   // text encoding setting is incorrect for the raw data bytes stored in it.
   Print textWave0[0]

   // Here is one way to fix the problem - convert the string variable to UTF-8.
   String textUTF8 = ConvertTextEncoding(text, textEncodingMacRoman,
                                         textEncodingUTF8, 1, 0)
   textWave0 = textUTF8                     // Store correct data

   // Print again - this time it prints correctly because the wave's
   // raw text bytes are consistent with the wave text encoding setting.
   Print textWave0[0]

   // Create the problem again so we can see another way to fix it.
   textWave0 = text                         // Store incorrect data

   // Here is another way to fix the problem - by correcting Igor's idea
   // of what text encoding the wave is using.
```