```
   // Tell Igor that the text wave stores text encoded as MacRoman.
   SetWaveTextEncoding textEncodingMacRoman, 16, textWave0

   // Print again - this time it prints correctly because Igor knows
   // that the wave contains MacRoman and so converts from MacRoman
   // to UTF-8 before sending the text to the history.
   Print textWave0[0]
End
```

Text Encoding Programming Issues

Igor6 and earlier versions of Igor stored text in "system text encoding". For western users this means Mac-Roman on Macintosh and Windows-1252 on Windows. For Japanese users it means Shift JIS on both platforms.

Igor now uses UTF-8, a form of Unicode, exclusively.

If you are an Igor programmer attempting to support Igor6, and if you or your users use non-ASCII text, the introduction of Unicode in Igor7 entails compatibility challenges.

If you fall in this category, we highly recommend that you abandon the idea of supporting Igor6 and later versions with the same procedure files. Instead, freeze your Igor6 code, and do all new development in a new fork. Here are the reasons for not attempting to support Igor6 and later versions with the same code base:

1. You will be unable to use new features without using ifdefs and other workarounds.
2. The use of ifdefs and workarounds will clutter your code and introduce bugs in formerly working Igor6 code.
3. You will have to deal with tricky text encoding issues, as explained in the next section.
4. Eventually Igor6 will be ancient history.

The alternative to supporting both Igor6 and later versions with the same procedure files is to create two versions of your procedures, one version for Igor6 and another for Igor7 and beyond. The benefits of this approach are:

1. You avoid cluttering your Igor6 code or introducing new bugs in it.
2. It will be much easier to write new code and it will be cleaner.
3. Your new code can use the full range of Unicode characters.

The costs of this approach are:

1. If you fix a bug in the Igor6 version of your code, you will have to fix it in the new version also.
2. New features of your code will be unavailable to Igor6 users.

The next section gives a taste of the text encoding issues raised by the use of Unicode starting with Igor7.

Literal Strings in Igor Procedures

Consider this function, which prints a temperature with a degree sign in Igor's history area:

```
Function PrintTemperature(temperature)
   Variable temperature

   // The character after %g is the degree sign
   Printf "The temperature is %g°\r", temperature
End
```