```
String units = "µs"
Printf "The unit prefix is %s\r", units[0]
```

This fails in Igor7 or later, and prints a missing character symbol, because the expression "units[0]" returns a single byte, but in UTF-8, "µ" is represented by two bytes. Therefore "units[0]" returns the first byte of a two-byte character and that is an invalid character in UTF-8.

If you are tinkering with the contents of a string, and if the string may contain non-ASCII text, you need to be clear when you want to go byte-by-byte and when you want to go character-by-character. To go byte-by-byte, you merely use sequential indices. At present, there is no built-in support in Igor for going character-by-character. See **Character-by-Character Operations** on page IV-173 for an example of stepping through characters using user-defined functions.

Automatic Text Encoding Detection

Techniques exist for attempting to determine a file's text encoding from the codes it contains. Such techniques are unreliable, especially with the kind of text commonly used in Igor. Consequently Igor does not use them.

Much of Igor plain text is procedure text. Procedure text is mostly ASCII, sometimes with a smattering of non-ASCII characters. The non-ASCII characters may be, for example, MacRoman, Windows-1252, or Japanese.

There is no reliable way to distinguish MacRoman from Windows-1252 or to reliably distinguish a procedure file that contains a smattering of non-ASCII western text from one that contains a smattering of Japanese.

For example, consider an Igor procedure file that contains just one Japanese two-byte character encoded in Shift JIS by the bytes 0x95 and 0x61. This can be interpreted as:

Shift JIS:	CJK UNIFIED IDEOGRAPH-75C5
MacRoman:	LATIN SMALL LETTER I WITH DIERESIS, LOWERCASE LETTER A
Windows-1252:	BULLET, LOWERCASE LETTER A

All three of these interpretations are possible and choosing among them is guesswork so Igor does not attempt to do it.

Shift JIS Backslash Issue

Because Igor now internally process all text as UTF-8, it must convert text that it reads from files that use other encodings. A special issue arises with Japanese text stored in Shift JIS format.

In Shift JIS, the single-byte code 0x5C is hijacked to represent the half-width yen symbol. In ASCII, 0x5C represents the backslash character. When Shift JIS text is converted to UTF-8, Igor's conversion software leaves 0x5C unchanged. Thus its interpretation changes from a half-width yen symbol to a backslash symbol.

In most cases this is the desired behavior because the half-width yen symbol is used in Japanese like the backslash in ASCII - to separate elements of Windows file system paths and to introduce escape sequences in literal strings.

If you are using a half-width yen symbol as a currency symbol then this conversion will be wrong and you will have to manually convert it to a half-width yen symbol by editing the text.

In UTF-8, the half-width yen currency symbol is represented by the code units 0xC2 and 0xA5. It can be entered in a literal text string as "\xC2\xA5".