

```
ShiftJIS = UTF8
```

The last statement triggers a fetch from the wave UTF8 and a store into the wave ShiftJIS. A fetch converts a text wave's data to UTF-8. In this case it is already UTF-8 so no conversion is done. A store converts the data to the text encoding of the destination wave - Shift JIS in this case. You wind up with the same characters in ShiftJIS and in UTF8 but not the same raw bytes because they are governed by different text encodings.

Igor does not do text encoding conversion when fetching from a wave or storing into a wave if the wave is marked as binary. For example:

```
// Make a wave to contain binary data
Make /O /T /N=1 Binary                                // UTF-8 by default
SetWaveTextEncoding 255, 16, Binary                      // Now binary
Binary = UTF8
```

The last statement's storing action does not do text encoding conversion because the destination wave is marked as binary. Thus Binary will wind up with the same bytes as in UTF8.

Now consider this:

```
Binary = ShiftJIS
```

Once again the last statement's storing action does not do text encoding conversion because the destination wave is marked as binary. You might think that Binary would wind up with the same bytes as in ShiftJIS but you would be wrong. The reason is that the fetching action triggered by evaluating the righthand side of the assignment statement converts the Shift JIS text in ShiftJIS to Igor's internal standard, UTF-8. Consequently the same bytes wind up being stored in Binary after each of the two previous statements.

The moral of the story is that, if you want to transfer binary data between text waves, make sure that both waves are marked as binary. It will also work if the source wave is marked as UTF-8 since the fetch conversion does nothing if the source is UTF-8.

## Manually Setting Wave Text Encodings

**NOTE:** Back up your experiment before fiddling with text encoding settings.

With the concepts of wave plain text elements, their text encoding settings, and text waves containing binary data in mind, we can now consider a general approach for setting wave text encodings. This is a task for advanced users only.

As a test case, we will assume that you created an experiment using Igor Pro 6.22 on Windows with English as the system locale. Consequently the experiment contains waves with all text encoding settings set to unknown. You then opened the experiment in Igor Pro 6.30, again on Windows with English as the system locale, and created additional waves. These added waves have text encoding settings set to Windows-1252 except for the text wave content element which is set to unknown. Finally, somewhere along the line, some text waves were created to store binary data, possibly by a package that you use and without your knowledge.

We now open the experiment in Igor7 or later and try to set all of the text encoding settings to correct known values using the SetWaveTextEncoding operation. The magic numbers used in the following commands are detailed below under **Text Encoding Names and Codes** on page III-490 and **SetWaveTextEncoding**.

```
// Mark any text waves containing binary data as such.
// 255 means "binary". 16 means "text wave content".
// /DF={root:,1} means "all waves in all data folders".
// /BINA=1 means "automatically mark text waves containing binary data".
SetWaveTextEncoding /DF={root:,1} /BINA=1 255, 16

// Mark any wave text elements currently set to unknown as Windows-1252.
// 0 means "unknown". 3 means Windows-1252. 31 means "all wave elements".
```