

## Chapter III-16 — Text Encodings

---

When Igor6, running on Macintosh, loads an experiment written on Windows, it converts text from Windows-1252 to MacRoman. The opposite occurs if, running on Windows, it loads an experiment written on Macintosh.

When Igor loads an Igor6 experiment written on Macintosh, it converts from MacRoman to UTF-8. If the experiment was written on Windows it converts from Windows-1252 to UTF-8.

### Asian Text Encodings

Asian languages have thousands of characters so it is not possible to map one character to one byte as in western text encodings. Consequently various multi-byte text encodings were devised for different Asian languages. Such systems are sometimes called MBCS for "multi-byte character system".

Shift JIS is the predominant MBCS text encoding for Japanese. It uses a single byte for all ASCII characters and some Japanese characters and two bytes for all other Japanese characters.

In Shift JIS, the codes from 0x00 to 0x7F have the same meaning as in ASCII except for 0x5C which represents a backslash in ASCII but a yen symbol in Shift JIS. The codes from 0xA1 to 0xDF represent single-byte katakana characters. The codes from 0x81 to 0x9F and 0xE0 to 0xEF are the first bytes of double-byte characters.

Chinese and Korean text encodings also use MBCS.

Unlike the situation for western text, Macintosh and Windows both use the same, or practically the same, text encodings for Japanese. The same is true for Traditional Chinese and Simplified Chinese.

The MacJapanese and Windows-932 text encodings are variants of Shift JIS. In this documentation, "Shift JIS" and "Japanese" mean "Shift JIS or its variants MacJapanese and Windows-932".

When Igor loads an Igor6 experiment written in Shift JIS it converts text from Shift JIS to UTF-8.

### Unicode

The proliferation of text encodings for different languages and even for the same language on different operating systems complicated the exchange and handling of text. This led to the creation of Unicode - a system for representing virtually all characters in all languages with a single text encoding. The first iteration of Unicode was released in the early 1990's. It was based on 16-bit numbers rather than the 8-bit bytes of previous text encodings.

In 1993 Microsoft released Windows NT, one of the first operating systems to use Unicode internally. Mac OS X, developed in the late 1990's and released in 2001, also uses Unicode internally. Most applications at the time still used system text encoding and the operating systems transparently translated between it and Unicode as necessary.

In the mid-1990's the developers of Unicode realized that 16-bits, which can represent 65,536 distinct numbers, were not enough to represent all characters. They extended Unicode so that it can represent over one million characters by using two 16-bit numbers for some characters. However nearly all of the commonly-used characters can still be represented by a single 16-bit number.

Over time most applications migrated to Unicode for internal storage of text. Igor7 was the first version of Igor that uses Unicode internally.

### Unicode Character Encoding Schemes

Unicode maps each character to a unique number between 0 and 0x10FFFF (1,114,111 decimal). Each of these numbers is called a "code point".

There are several ways of storing a code point in memory or in a file. Each of them is called a "character encoding scheme". Each character encoding scheme is based on a "code unit" which may be a byte, a 16-bit value or a 32-bit value.