

Overview

Starting with Igor Pro 7, Igor stores text internally as UTF-8, a Unicode text encoding format. Using Unicode improves Igor's interoperability with other software and gives you easy access to a wide array of characters including Greek, mathematical and special characters.

Previous versions of Igor used non-Unicode text encodings such as MacRoman, Windows-1252, and Shift JIS (Japanese), depending on the operating system you were running. Consequently Igor needs to do text encoding conversions when opening files from earlier versions.

If a file contains accented characters, special symbols, and other non-ASCII text, it is not always possible to get this conversion right. You may see incorrect characters or receive "Unicode conversion errors" or other types of errors.

Text encoding is not an issue with files that contain only ASCII characters as these characters are represented using the same codes in UTF-8 as in other text encodings.

This chapter provides information that a technically-oriented Igor user can use to understand the conversions and to deal with issues that can arise. To understand these issues, it helps to understand what text encodings are and how they are used in Igor, so we start with an overview.

Text Encoding Overview

A text encoding is a mapping from a set of numbers to a set of characters. The terms "text encoding", "character encoding" and "code page" represent the same thing.

In most commonly-used text encodings, text is stored as an array of bytes. For example, the text "ABC" is stored in memory as three bytes with the numeric values 0x41, 0x42 and 0x43. (The 0x notation means the number is expressed in hexadecimal, also called base 16.)

In the 1960's the ASCII text encoding was defined. It specifies the mapping of 128 numbers, from 0x00 to 0x7F, to characters. ASCII defines mappings for:

- Unaccented letters (A-Z and a-z)
- Numerals (0-9)
- Common punctuation marks (comma, period, dash, question mark ...)
- Other printable characters (space, brackets, slash, backslash, ...)
- Control characters (carriage-return, linefeed, tab, ...)

The "ABC" example above uses the ASCII codes for A, B and C.

Over time, other text encodings were created to permit the representation of characters not supported by ASCII such as:

- Accented western letters
- Asian characters
- Mathematical symbols

Hundreds of text encodings were created for encoding characters used in different languages. Even for a given language, different operating systems often adopted different text encodings.

In ASCII, each character is represented by a 7-bit code which is typically stored in a single 8-bit byte with one bit unused. 7 bits can represent 128 different characters. To represent more characters, other text encodings use 8 bits per character, multiple bytes per character, 16-bit codes or 32-bit codes.

We use the term "byte-oriented text encoding" to distinguish text encodings that represent characters using one byte or multiple bytes per character from those that use 16-bit or 32-bit codes.