Representing Scripts on the Macintosh

Worldwide system software makes it possible to represent many scripts and languages on the Macintosh. The Macintosh Script Management System extends the Macintosh computer's text-manipulation capabilities beyond Roman scripts. Character representation and the keyboard are the first components to be considered when attempting to represent any script on a Macintosh computer.

Character Set Encoding

Character set encoding refers to the numeric codes that represent the characters of a script in memory. The character set encoding for a script determines the behavior of many of the features of the script, including sorting and composition rules for drawing and measuring. Therefore, the character set encoding is fixed; it cannot be changed without significant consequences. For example, features such as sorting depend on the fact that the coding does not change.

Most scripts fit within the limits set by the size of a byte, with up to 256 distinct characters. Scripts with ideographic characters, such as Chinese, Japanese, and Korean, need more than 256 distinct characters.A variety of solutions have been implemented for scripts that require 2-byte codes for computer storage in addition to or in place of the 1-byte codes that are sufficient for Roman scripts. Proper use of the **Script Manager** routines permits your application to run without knowing whether 1-byte or 2-byte codes are being used, as long as it has been written to allow the possibility of 2-byte codes.

> **Warning:** Typically, ideographic scripts use a mixture of single-byte and double-byte encodings to represent characters; therefore, you cannot use the terms *byte* and *character* interchangeably.

> **Note:** Currently, every different character set encoding does not have a different script code. For example, the Symbol font is in the Roman range but has a different character assignment. Before system software version 7.0, the traditional and simplified Chinese systems used different character codings, but had the same script code (2). With system software version 6.0.5, these systems have been assigned separate script codes: (2) and smSimpChinese (25).

Character Input

Character input is often more complicated than simply providing a keyboard layout. Ideographic scripts such as Japanese cannot simply use a larger keyboard or multiple dead keys for effective input. The sheer number of characters demands a more complex solution, such as providing ways to transcribe phonetic text into ideographic text. Most ideographic script systems provide for the complex parsing of phonetic sequences and character clusters.

Composition Rules

Each script system contains composition rules that determine the behavior (that is, the visual appearance) of text when it is drawn, measured, or edited. These intricate rules also provide for other features of the script, such as

determining when a sequence of characters forms a word or whether a byte is a single character or part of a double-byte character.

Text Manipulation

With a flexible operating system, most script features are implemented transparently. Usually, your application does not need to know that its dialog boxes can accept Japanese text. However, if your application depends more heavily on features of the language, you need access to information that varies with the script.

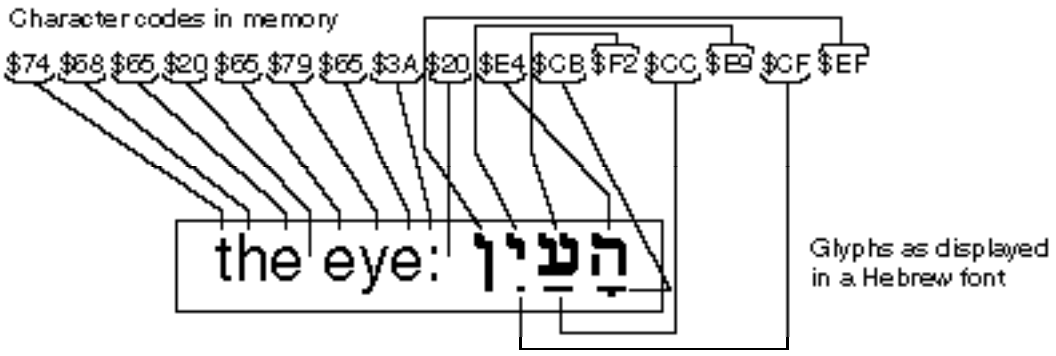
For example, to perform word selection and **word wrap** (the automatic continuation of text from the end of one line to the beginning of the next without breaking in the middle of a word), your application may need routines to determine the boundaries of words in the script. In Roman scripts, you can determine word boundaries fairly easily: spaces delimit words that are not otherwise delimited (for example, by punctuation). Other scripts, such as Japanese and Thai, do not use characters such as spaces to delimit words. Word boundaries are not well defined, and native writers of the language may not agree on where particular word boundaries occur.

Because of differences in the treatment of uppercase and lowercase characters and diacritical marks, your application may need routines and tables to perform case conversion and, when sorting is to be done, to strip diacritical marks. The Macintosh Script Management System supplies such routines; they are described briefly in **Manipulating Text** and **Converting Case and Stripping Diacritical Marks** and in *Macintosh Worldwide Development: Guide to System Software*.

In addition, your application may require routines to determine whether a byte represents a single character or is part of a double-byte character, or to highlight bidirectional text (for example, Arabic mixed with English). To allow applications to function independently of scripts, the Macintosh Script Management System provides such routines; they are described briefly in **Drawing and Editing Text** under **Manipulating Text** and in *Macintosh Worldwide Development: Guide to System Software*.

Text Rendering

The process of displaying characters that are stored in memory is called **text rendering. Backing-store order** refers to the order in which character codes are stored in memory. In general, characters for a given script are stored in writing order, that is, the order in which someone would set the characters down on paper. This may be different from the **display order,** that is, the left-to-right order in which characters are *drawn* on a device by **QuickDraw**. The **Script Manager** then handles differences between backing-store order and **QuickDraw** display order. For example, Hebrew characters appear on the screen so that the glyph corresponding to the first character in the string actually appears on the right of the string (see the Figure below). In another example, when diacritical marks are stored as separate overlapping characters, they are typically stored *after* the base character. Writing order is very similar to **phonetic order**, that is, the order in which the characters are pronounced, but the two differ in certain circumstances. In some cases, the phonetic order is not well defined, as with diacritical characters.



Backing-store and display order