Technical Note TN1152

JIS Keyboard Support in Mac OS 8

CONTENTS

This technote describes the mechanism introduced in Mac OS 8 to support JIS (Japanese Industrial Standards) keyboards. Developers designing input methods or applications that rely on keyboard layout information should read this technote.

Updated: [Feb 22 1999]

JIS Keyboards

Macintosh keyboards sold in Japan come in two flavors: US and JIS. As the name suggests, US keyboards are identical in layout to keyboards sold in the United States. JIS keyboards are unique to Japan and feature additional key caps designed for input method functions. For example, the iMac keyboard has the following JIS layout in Japan.

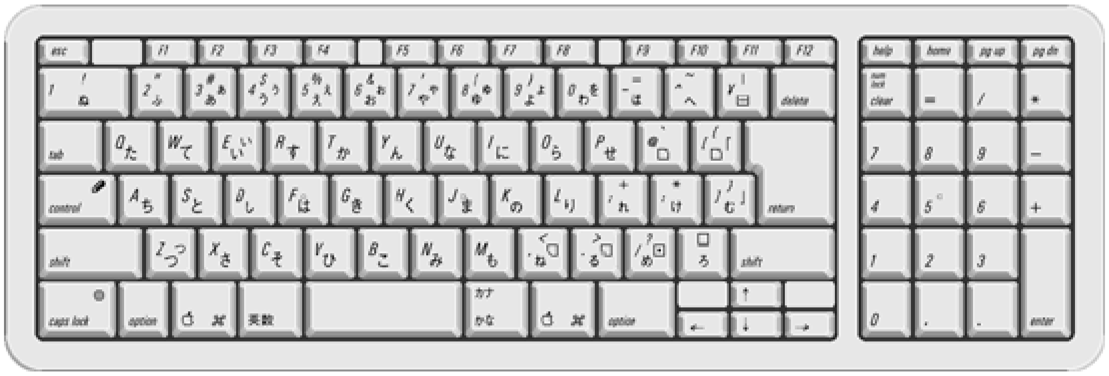

Figure 1 - The iMac keyboard in Japan

The World Before Mac OS 8

Japanese versions of the Mac OS contain additional resources to accommodate US and JIS keyboard layouts. Prior to Mac OS 8, this took the form of six extra keyboard-layout resources.

Name	Script	How Japanese characters are typed	Keyboard layout
Kana	Japanese	Kana	US
Kana - JIS	Japanese	Kana	JIS
Romaji	Japanese	Romaji	US
Romaji - JIS	Japanese	Romaji	JIS
U.S.	Roman	N/A	US
Roman - JIS	Roman	N/A	JIS

Table 1: Keyboard-layout resources (resource type `'KCHR'`) in Mac OS 7.x

Users were required to configure the correct keyboard-layout resource for each installed script by selecting it in the Keyboard control panel. Since only keyboard-layout resources that belong to the current script are displayed in the control panel, and a Japanese system typically contains two scripts (Japanese and Roman), at least two operations were necessary to specify the type of keyboard attached to the computer.

For example, in the case of a JIS keyboard, the user would first select "Roman - JIS" in the Keyboard control panel.

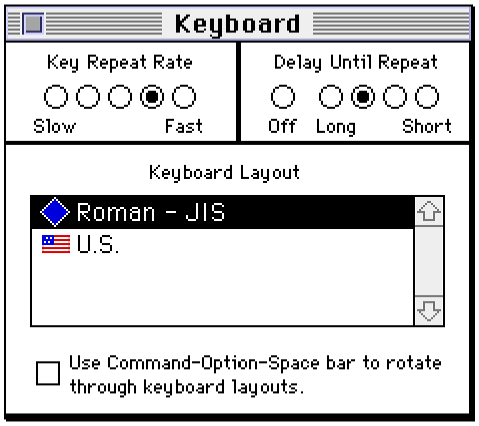

Figure 2 - Selecting "Roman - JIS" in the Keyboard control panel

Next, the user would switch the active script to Japanese (from the script menu), and select "Kana - JIS" or "Romaji -JIS" from the Keyboard control panel, depending on their preferred method of typing.

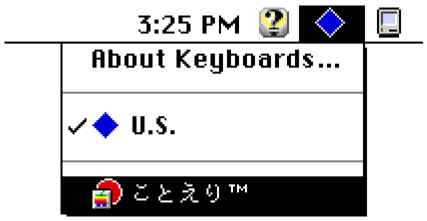

Figure 3 - Switching the active script to Japanese

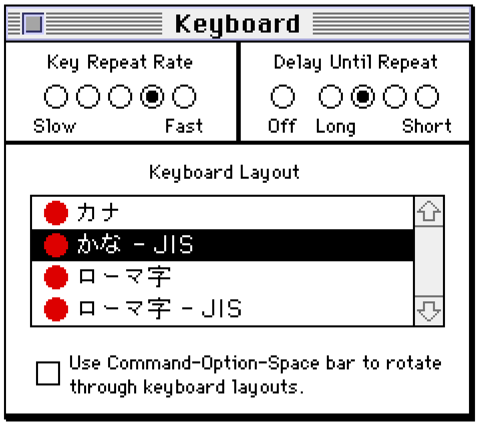

Figure 4 - Selecting "Kana - JIS" in the Keyboard control panel

Note:
While keyboard selections are persistent across restarts in all versions of the Mac OS, the active script is reset when the computer starts up.

Selecting an incorrect keyboard type for either Japanese or Roman scripts would cause strange behavior. For example, selecting a non-JIS layout from the Keyboard control panel when a JIS keyboard was in fact connected would result in certain keys not generating `keyDown` events. This is because JIS keyboards feature additional keys that are not present on standard US keyboards; in the standard US keyboard-layout resource, non-existent keys are mapped onto nil character codes which do not generate `keyDown` events. To make matters worse, some keys actually map onto different characters altogether.

Another problem prior to Mac OS 8 was the lack of a reliable method to determine whether a keyboard's layout was JIS or not. Input methods circumvented this by maintaining static tables of keyboard IDs that were known to be JIS. By comparing the `KbdType` low-memory global variable (`LMGetKbdType`) to keyboard IDs in their internal tables, input methods could determine if an arbitrary keyboard was JIS or not. Needless to say, every time Apple shipped a JIS keyboard with a new ID, these hard-coded input methods would break and require updating.

As if this weren't confusing enough, most input methods bypassed the Keyboard control panel altogether and implemented their own Kana and Romaji settings. So, in fact, it didn't make a difference whether "Romaji / Romaji-JIS" or "Kana / Kana-JIS" was selected in the Keyboard control panel.

Back to top

The World After Mac OS 8

To remedy the above problems, Apple made the following improvements in Mac OS 8:

- **New Key-Remap Resources**

  Instead of requiring users to adjust settings in the Keyboard control panel according to the type of keyboard attached, key-remap resources (resource type `'itlk'`) were added to support JIS keyboards in Mac OS 8.0. Since the keyboard type is handled at the key-remap resource level, keyboard layout settings no longer need to be visible to the user. As a result, JIS-specific keyboard layout resources became redundant and were removed.

Name	Script	How Japanese characters are typed	Keyboard layout
Kana	Japanese	Kana	Any
Romaji	Japanese	Romaji	Any
U.S.	Roman	N/A	Any

Table 2: Keyboard-layout resources (resource type `'KCHR'`) in Mac OS 8

  By keeping keyboard-layout resources independent of the keyboard type, just three resources are required to support both JIS and non-JIS keyboards.

- **KBGetLayoutType**

  A new API, `KBGetLayoutType`, was introduced in Mac OS 8.0 to return the layout type (ANSI, JIS, or ISO) of the current

- **A new Keyboard control panel**

  With Mac OS 8.0, Apple introduced a new Keyboard control panel. In addition to user interface consolidation and refinements, the new control panel hides keyboard-layout resources for two-byte scripts from the user. This is possible because keyboard layouts are now handled by key-remap resources and responsibility for specifying how Japanese characters are typed (Kana or Romaji) now lies with the input method.

The rest of this technote describes each of these improvements in detail.

New Key-Remap Resources

When a key is pressed, the keyboard generates a raw key code which is mapped onto a virtual key code using a key-map resource (resource type `'KMAP'`). The Event Manager uses the appropriate keyboard-layout resource to map this virtual key code onto a more useful character code, which is later passed on to the application in a `keyDown` event. The key-remap resource (resource type `'itlk'`) adds an extra conversion layer, mapping the virtual key code generated by the key-map resource onto another virtual key code when it is handled inside `KeyTranslate`. For details, see *Inside Macintosh: Text*, Appendix C, C-16

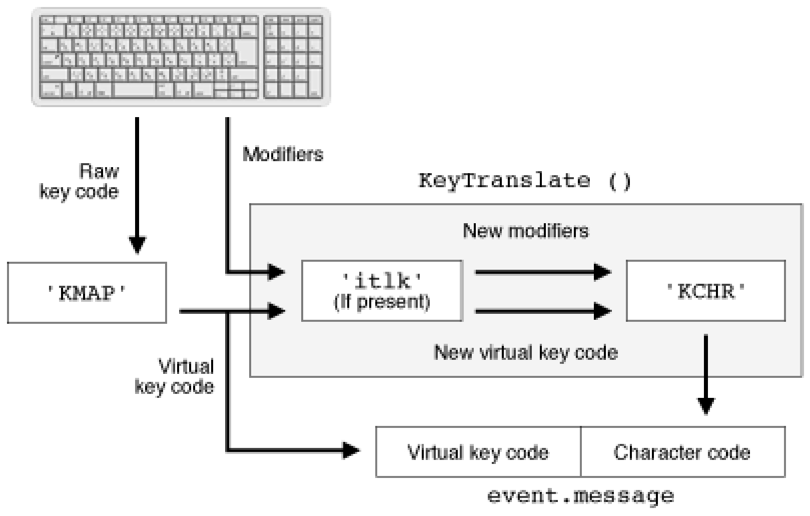

Figure 5 - The general flow of a key event with intervention by a key-remap resource.

What happens without the key-remap resource?

When a key is pressed, the key-map resource (resource type `'KMAP'`) whose ID matches the keyboard's device ID is used to convert the raw key code into a virtual key code. For example, pressing the "@" key on an JIS Apple Keyboard II (ID = 22) will generate a virtual key code of `0x21`. If the currently selected keyboard-layout resource is "Roman - JIS", `KeyTranslate` will map `0x21` onto `0x40`, which is the correct character code for the "@" symbol.

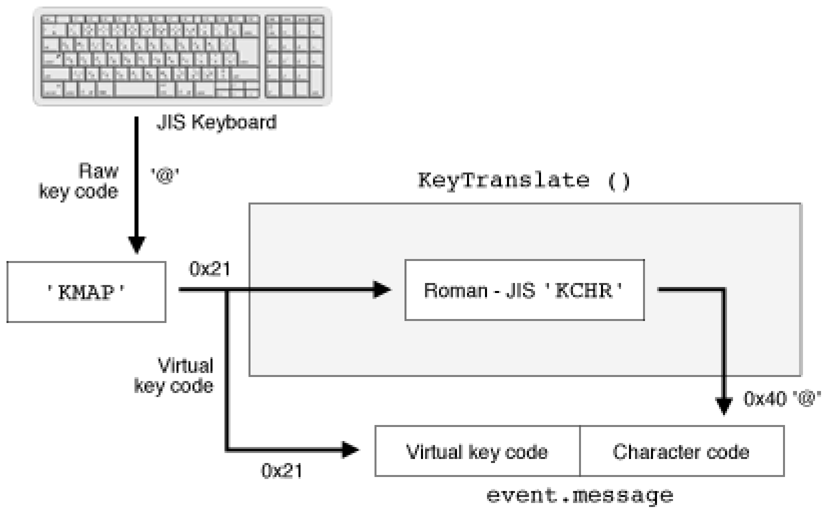

Figure 6 - Pressing "@" on a JIS keyboard with "Roman - JIS" selected.

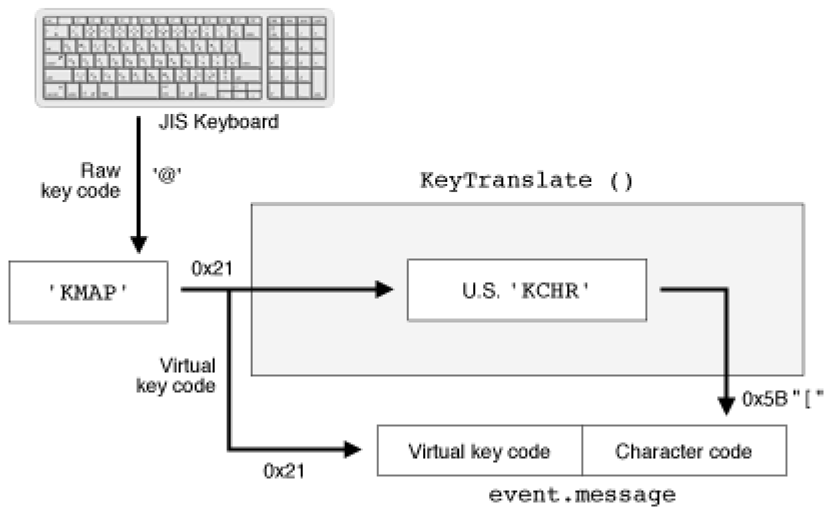

Figure 7 - Pressing "@" on a JIS keyboard with "U.S." selected.

In order to generate the correct character code for "@" (`0x40`) on a US keyboard with the "U.S." `'KCHR'` selected, a virtual key code of `0x13` needs to be generated with the Shift Key modifier flag set.

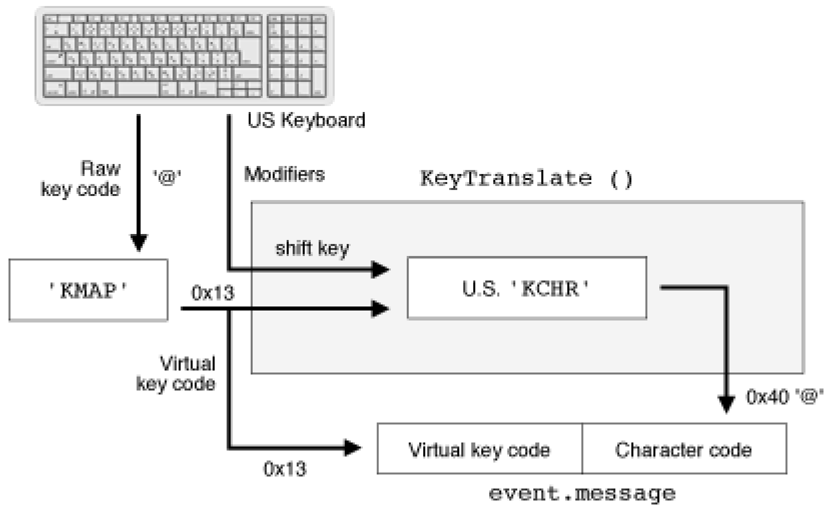

Figure 8 - Generating "@" on a US keyboard with "U.S." selected.

In order to generate the correct character code for "@" (`0x40`) on a JIS keyboard with the "U.S." `'KCHR'` selected, a key-remap resource is needed to map the virtual key code onto `0x13` and set the Shift Key modifier flag.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.