

Chapter 4

Cultural Differences

In this chapter I will give an introduction about the efforts, that have to be made to localize, in this case to japanize, software. I will show the main differences and in the next chapters I will introduce the reader to possible solutions (starting from 163).

4.1 Introduction

The general difference between the European/American culture and the Japanese culture is not only that Japan is an Asian country. It is, based on the different culture and religions, also the way they do the things the do. This is affected not only by the Japanese ”modus operandi”, it is also effected by the environment, which is given by their tradition, language and writing. Some people refer to this as ”Japanese language problems” (see [31]), but I think that only we got the problems and not the Japanese. That means that we have to modify our software if we want to sell it on the Japanese market.

So let us have a closer look on the requirements for the localization / internationalization of a software product for the Japanese market (see [2, 1]) :

- Numbers, the representation of numbers, decimal separator, ...
- Currency, representation of the currency, average number of digits, decimal separator.

- Date convention, which date convention is used. How is a date represented? Are there other calendars then the Gregorian calendar used.
- Character set, the use of alphabet & numerals (the roman alphabet is called Romaji), Hiragana, Katakana and Kanji; special characters, double & single byte character sets. Special requirements like :
 - input methods (e.g., Keyboard, Pen) for Hiragana, Katakana and Kanji.
 - storing of characters as 1 or 2 Byte code (SBCS, DBCS)
 - displaying and printing characters
 - different standards for representation of character sets
- paper size
- units, etc.
- Manual, On-line help, support and telephone hotline should be in Japanese, e.g., a local office in Japan
- programming language (application) should be able to handle the Japanese character set
- To provide the right service for Japanese customers

In order to do this you need a knowledge about the cultural specialities which apply in the Japanese language environment. I will give you an overview on the next couple of pages.

4.2 Japanese Character Sets

When the Japanese started to build the first computer systems back in the 1960s they followed the example given by the US computer industry (see [3]). This first computer types were built like US models and used the same type of character set called ASCII (see figure 4.1, page 65). It was, for the stage of technology at this time, quite difficult to handle the specific Japanese character sets like the syllable alphabets Hiragana and Katakana or the thousands of ideographic Chinese characters called Kanji. This type of computer systems were not very sophisticated for the Japanese computer user. As mentioned before it is possible to write Japanese names and addresses in Romaji but it makes it very difficult to read a Japanese text, written entirely in Romaji. The Japanese use Kanji characters which are ideographic characters. This means that Kanji characters (or a combination of them) represent an idea, meaning or thought. It would be possible to write down the pronunciation but this is quite difficult because there are different systems to transliterate a Japanese Yomi (pronunciation) to the roman (Romaji) characters (e.g., Hepburn, Nipponsiki; see figure 4.2 starting from page 69). A Kanji character could have so different spellings depending on chosen transliterating system. Besides that a Kanji character could have different Yomi's (see c in figure 4.38 on page 125) or a Yomi could have different Kanji characters (see b, meiji, in figure 4.38 on page 125) depending on the meaning. This makes it very difficult for Japanese to use only Romaji.

4.2.1 ASCII and Katakana

The next step of the Japanese computer industry was to adapt the Katakana alphabet to the computer character set. This had some reasons, like :

- only a limited number of characters
- useful to express foreign and Japanese " words "
- relatively easy to implement on a computer
- depending on their shape they are easy to print or display

- does not require a Font End Processor (see page 115) for the input of the characters. It is easy to realize with a new level of characters on a standard keyboard (e.g., see figure 4.7 on page 74, [28]).

To implement Katakana characters was the easiest way to add, at least some, Japanese capabilities to a computer system. It has only a limited number of characters (which where fitting in the space above 127 decimal, $7F_{Hex}$) so that it was possible to use the 7(8)-bit architecture. You could get along without major changes (see figure 4.9 and 4.10 on page 76 and 77). This one byte code (Single Byte Character Set, SBCS) is called JIS X0201-1989 (the name changed, march 1987, from the old name JIS C6220-1976) and describes an enhanced ASCII character set which includes Katakana characters. The use of this Katakana character set has the advantage that, through the limited number of characters, it is possible to use a standard keyboard and shift between ASCII and Katakana input (see figure 4.7 and 4.8 on page 74 and 75, [28]). This keyboard layout is naturally defined by an own standard called JIS X6002-1984 (or the predecessor JIS C 6233-1980). On the keyboard layout appear the 52 small and capital Roman alphabet characters, ten numerals, 32 special characters (like !, \$, &, @, +, -, etc.), 8 Japanese special characters, 17 control characters (like CR, LF, ETX, DEL, ESC, ...) and 55 Katakana ¹ characters. A standard definition does, unfortunately, not mean that everybody has to follow this definition. This causes that there are different keyboard layouts available.

7-bit JIS

This code exists in a 7-bit and a 8-bit version. The difference between the versions is that in the 7-bit (from 00 to $7F_{Hex}$) version a Shift In (SI, $0F_{Hex}$, sometimes called Kanji In (KI)) and a Shift Out (SO, $0E_{Hex}$, sometimes called Kanji Out (KO)) character is used to shift between the ASCII and Katakana code table. This means that the system starts printing ASCII characters until it runs over a SO. All following characters are printed as Katakana characters. This stops when the system finds a SI, which switches back from Katakana mode to ASCII mode. The use of a SI and SO character to switch between code tables could cause some problems which I will later explain more in detail.

¹Which is actually not the complete set of Katakana characters

8-bit JIS

This problem does not occur when your system is able to use the 8-bit version of the JIS X0201-1989. In this case the system must be able to work with 8-bit characters (called "8-bit clean", which was not always possible, e.g., in early implementations of UNIX, they sometimes used the highest bit as a parity bit). With the 8-bit version you do not have to use the SI and SO characters to switch between the ASCII and Katakana code table. The Katakana characters are just located in the, former unused, area above $7F_{Hex}$.

The use of this area could cause some problems when you work with, e.g., American IBM PC software. The IBM PC has a totally different codetable in the area between $7F_{Hex}$ and FF_{Hex} . If you start using foreign software it could (or definitely will) happen that a screen mask looks very funny because instead of line-elements Katakana characters appear. This results in a nearly ASCII compatible character set. In the 7-bit version one of the code tables is nearly compatible to ASCII. In the 8-bit version the area below $7F_{Hex}$ is nearly compatible. The only difference, which makes the character set just nearly compatible, is that the backslash (\backslash , $5C_{Hex}$) is replaced by the Yen symbol (see picture d in figure 4.32, page 114). The second replacement is the tilde (\sim , $7E_{Hex}$) which is replaced by the overline ($\bar{}$). All other characters correspond to their ASCII equivalent.

These Katakana characters have the same size as an ASCII character. The Katakana characters in this size are called half-width Katakana (in Japanese Hankaku). Still this was not a very sophisticated solution for the early Japanese computer users. The lack of Kanji characters was one of the important points which made them start to think about how to integrate Kanji characters into computer systems.

4.2.2 Development of Kanji Character Sets

To understand the following development of the Kanji character code set we have to have a look on non-electronic (see [8]) character sets which were used to define the JIS C 6226-1978. Which was leading towards JIS X0208-1990, the standard today.

The Japanese have about 40000 to 60000 "known" Kanji characters. The problem with that is that nobody is able to remember all of them. The Ministry of Education

started to restrict the number of Kanji characters for the use in education. Today a Japanese student learns about 2000 Kanji characters.

The historical development of the standard was started with the table of sanctioned Kanji characters for education. This first table was called Toyo Kanji and contained, in 1946, 1850 Kanji characters. This table was replaced by the Joyo Kanji character table in 1981. This table contains now 1945 Kanji character (see figure 4.11, page 78). The other tables which were used to form the standard character set are the Gakushu Kanji (replaces the older Kyoiku Kanji table with 881 Kanji characters) with 1006 characters (increased in 1992 from 996 Kanji characters) and the Jinmei-yo Kanji character table which has since 1990 284 characters (increase from 85 characters in 1946 to 112 characters in 1976 then to 166 characters in 1981). An interesting fact is that Gakushu Kanji is a subset of Joyo Kanji (see [8]).

Double Byte Character Sets

This non-electronic character sets were used to define the actual DBCS character set standard JIS X0208-1990. Besides the Kanji, Hiragana (83) and Katakana (86) characters the standard includes alphanumeric characters (10 numerals, 52 Roman characters), special characters (147 symbols), Greek (48) and Russian (66) letters and rule line elements (32). During the years there were some changes (X208 was first established 1978, the first change occurred 1983, the actual version is from 1990) which added some new Kanji characters, changed the shape of some characters or a change in the positions of some characters has taken place. Today this standard contains two levels with 2965 characters in level 1 and 3388 characters in level 2. In 1990 the JSA introduced a supplementary DBCS character set which is called JIS X0212-1990 (sometimes referred as JIS level 3) with additional 6067 characters. In addition to 5801 Kanji characters this standard contains 21 special characters and 245 Latin (Roman), Cyrillic and Greek characters (mostly with diacritical marks, characters like, e.g., German Umlauts, the French, Spanish or Danish special characters). This leaves us with a total number of 12156 standard characters, divided into three levels. Regarding the fact that JIS X0212-1990 is a very young standard the most systems use only the characters defined by the JIS X0208-19XX standard (see [25], [26], [27]). Nevertheless that this tremendous number of characters needs a lot memory. It is also impossible to represent this characters by using a SBCS. In order to represent this

huge number of characters we need at least a Double Byte Character Set (DBCS). In a standard 7 (or 8) bit environment we could use a character set which contains 127 (or 255) characters. This is enough to carry a standard ASCII character set and some national extensions, but it is not big enough to handle thousands of ideographic Kanji characters. In order to handle the tremendous number of characters we have to extend the number of bits which hold the character information. In a 7-bit environment a logical step is to use two 7-bit bytes (14-bits) to hold the information this would give us the possibility to store up to 2^{14} (16384) characters. If we use two 8-bit bytes we are able to store up to 2^{16} (65536) characters. The arising problem is how to distinguish between SBCS characters and DBCS characters. In order to stay compatible with the old SBCS character set you have to find a solution to determine if the actual byte is a SBCS character or belongs, as a part of it, to a DBCS character.

Shifting between SBCS and DBCS

Again, as mentioned above, it is possible to use the Shift In / Out mechanism to distinguish between SBCS and DBCS. This is quite useful in a 7-bit environment. Also it could be used in an 8-bit environment. Another possibility, in an 8-bit environment, is to use the MSB (Most Significant Bit) as a flag to show that this byte is a SBCS character (MSB = 0) or a part of a DBCS character (MSB = 1). A SBCS could look, in binary representation, like 0XXXXXXX and a DBCS would look like 1XXXXXXX 1XXXXXXX. Today the most large or medium sized systems uses a SI/SO (or KI/KO) sequence to switch between SBCS and DBCS characters. There is a recommendation from the JSA for this SI/SO sequence, but unfortunately the most hardware vendors have chosen different SI/SO sequences (usually between one and three bytes). Some examples for the SI/SO (KI/KO) sequences for New-JIS, Old-JIS, NEC-JIS, Lets-J, JBIS, JEF and IBM EBCDIC you will find in figure 4.1 (on page 66, [8], [9]).

Sometimes there are two different SI/KO sequences. One sequence switches back to the JIS-Roman character set. The other SO/KI sequence switches back to the ASCII character set.

Not only the SI/SO (KI/KO) sequences differ between the different implementations of a Kana/Kanji character set, also the location in the matrix which is defined by

the two bytes. Moreover some companies (like IBM) even do not use the JIS defined standard (for some examples see table 4.2 on page 67).

Starting from page 79 you will find the two byte matrices from different Kana/Kanji character sets. If you have a closer look on these matrices you will recognize that all vendors placed the JIS area or the extension area at different places. Although if the matrices are on the same place it does not mean that the same Kanji character appears at the same place. In the Japanese PC world the Shift JIS is the standard for the character set. This version of the JIS character set was moved to a different location (see figure 4.18 on page 85) because on this location it was possible to use the old 7-bit character set and the DBCS without a SI/SO (or KI/KO) sequence. In Shift JIS all 7-bit characters (SBCS) have the MSB set to 0 and look like 0XXXXXXX. If the MSB is set to 1 the byte is a part of a DBCS character (it looks like 1XXXXXXX 1XXXXXXX). An advantage of the Shift JIS design is that it is very easy to convert a JIS DBCS code to the corresponding Shift JIS DBCS code. To do this you could use the following formula ([7]) :

Shift JIS and JIS

SJIS is the two byte representation of the Shift JIS code and JIS the two bytes of the JIS code. SJIS₁ is the first byte and JIS₂ is the second byte of the code. The value for the bytes is between 00_{Hex} and FF_{Hex}

```

SJIS1 = (JIS1 - 21Hex) / 2 + 81Hex
if SJIS1 ≥ 9FHex then JIS1 = JIS1 + 40Hex
if odd(JIS1) then begin
  SJIS2 = JIS2 - 21Hex + 40Hex
  if (SJIS2 ≥ 7FHex) then SJIS2 = SJIS2 + 1
end
else SJIS2 = JIS2 - 21Hex + 9FHex

```

The Shift JIS is mainly used in the PC world and in few workstations. The most vendors offer conversion routines between their own code set and JIS and Shift JIS.

Another fact about the different character sets is that the user defined characters are placed on different locations in the two byte matrices. The number of the, so

called Gaiji characters, differs in each of the vendor's implementations. These Gaiji characters are needed because some Japanese names are written with "nonstandard" Kanji characters. If an, e.g., insurance company wants to print an invoice with the name of the customer it is common practice to use a user defined Gaiji character for this purpose (when the name of the customer contains a Kanji character which is not available as a JIS standard character).

4.2.3 Japanese Character Set Mess

When Fujitsu started 1978 ([9]) to introduce their main frame implementation of Japanese language processing nearly every vendor has introduced a different DBCS character set (some even more than one DBCS, see table 4.3 on page 68). Some European companies used an even more outer space approach, by designing totally incompatible Japanese character sets. Actually does this policy not increase the chances for selling software.

Even if all of them support JIS or Shift-JIS via conversion routines none of the systems is compatible to an other system. This leaves us with some problems for the japanization of foreign software. For each hardware platform you have to check the implementation and make (even slight) changes to adapt to this vendors' platform and DBCS character set.

Column ----- Row	00	01	02	03	04	05	06	07
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

ASCII Alphanumeric (Part 1)

Figure 4.1:

Standard	Kanji In	Kanji Out (JIS-Roman)	Kanji Out (ASCII)
New-JIS (1983)	ESC \$ B	ESC (J	ESC (B
Old-JIS (1978)	ESC \$ @	ESC (J	ESC (B
NEC-JIS (1978)	ESC K	ESC H	n/a
Lets-J	93 _{Hex} F0 _{Hex}	n/a	93 _{Hex} F1 _{Hex}
JBIS	2B _{Hex} (SOK)	n/a	2C _{Hex} (EOK)
IBM EBCDIC DBCS	0F _{Hex}	n/a	0E _{Hex} †
JEF	28 _{Hex}	n/a	29 _{Hex} †
JIS X0201-1976	0E _{Hex} SO	n/a	0F _{Hex} SI
JIS X0202-1984	ESC (I ‡		

Table 4.1: Different SO/KI and SI/KO codes

†= switches back to EBCDIC

‡= switches to half-width Katakana

Company	JIS	Extension
IBM (Host)	No	EBCDIC, IBM Basic Character set (3572 characters), IBM Extended Character set (3483 characters), IBM user (free) region (4370 characters)
IBM (PC)	Yes †	1880 User defined characters
AX Consortium	Yes †	448 User defined Hankaku characters, 752 User defined Zenkaku characters and 1024 Zenkaku characters defined in hardware
JEF (Fujitsu)	Yes	EBCDIC, 4039 Extended Kanji characters, 1083 Extended Non-Kanji characters, 3102 User defined characters
DEC	Yes	DEC JIS Extension up to 9621 User defined characters
LETS-J	Yes	Yes ‡
JBIS	Yes	Yes ‡
Hitachi	Yes	Yes ‡
NEC	Yes	Yes ddag

Table 4.2: Support of JIS standard character set

†= Shift JIS (see figure 4.18 on page 85)

‡= No information provided

Vendor	DBCS Implementation
Fujitsu	JEF
	JEF II
Hitachi	KEIS
NEC	JIPS
UNISYS	LETS-J
	JBIS
IBM	EGCS (old)
	DBCS (SAA)
PC-World	Shift-JIS
UNIX	EUC

Table 4.3: Different DBCS Implementations

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
a	a	あ	ア
i	i	い	イ
u	u	う	ウ
e	e	え	エ
o	o	お	オ
ka	ka	か	カ
ki	ki	き	キ
ku	ku	く	ク
ke	ke	け	ケ
ko	ko	こ	コ
sa	sa	さ	サ
<u>shi</u>	<u>si</u>	し	シ
su	su	す	ス
se	se	せ	セ
so	so	そ	ソ
ta	ta	た	タ
<u>chi</u>	<u>ti</u>	ち	チ
<u>tsu</u>	<u>tu</u>	つ	ツ
te	te	て	テ
to	to	と	ト
na	na	な	ナ
ni	ni	に	ニ
nu	nu	ぬ	ヌ
ne	ne	ね	ネ
no	no	の	ノ

a)-1 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
ha	ha	は	ハ
hi	hi	ひ	ヒ
<u>fu</u>	<u>hu</u>	ふ	フ
he	he	へ	ヘ
ho	ho	ほ	ホ
ma	ma	ま	マ
mi	mi	み	ミ
mu	mu	む	ム
me	me	め	メ
mo	mo	も	モ
ya	ya	や	ヤ
yu	yu	ゆ	ユ
yo	yo	よ	ヨ
ra	ra	ら	ラ
ri	ri	り	リ
ru	ru	る	ル
re	re	れ	レ
ro	ro	ろ	ロ
wa	wa	わ	ワ
wo	wo	を	ヲ
n	n	ん	ン

a)-2 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
ga	ga	が	ガ
gi	gi	ぎ	ギ
gu	gu	ぐ	グ
ge	ge	げ	ゲ
go	go	ご	ゴ
za	za	ざ	ザ
<u>ji</u>	<u>zi</u>	じ	ジ
zu	zu	ず	ズ
ze	ze	ぜ	ゼ
zo	zo	ぞ	ゾ
da	da	だ	ダ
<u>ji</u>	<u>di</u>	ぢ	ヂ
<u>zu</u>	<u>du</u>	づ	ヅ
de	de	で	デ
do	do	ど	ド
ba	ba	ば	バ
bi	bi	び	ビ
bu	bu	ぶ	ブ
be	be	べ	ベ
bo	bo	ぼ	ボ
pa	pa	ぱ	パ
pi	pi	ぴ	ピ
pu	pu	ぷ	プ
pe	pe	ぺ	ペ
po	po	ぽ	ポ

a)-3 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
kya	kya	きゃ	キャ
kyu	kyu	きゅ	キュ
kyo	kyo	きょ	キョ
kwa	kwa	くわ	クワ
<u>sha</u>	<u>sy</u> a	しゃ	シャ
<u>shu</u>	<u>sy</u> u	しゅ	シュ
<u>sho</u>	<u>sy</u> o	しょ	ショ
<u>cha</u>	<u>ty</u> a	ちゃ	チャ
<u>chu</u>	<u>ty</u> u	ちゅ	チュ
<u>cho</u>	<u>ty</u> o	ちょ	チョ
nya	nya	にゃ	ニャ
nyu	nyu	にゅ	ニュ
nyo	nyo	にょ	ニョ
fa	fa	ふぁ	ファ
fi	fi	ふぃ	フィ
fe	fe	ふぇ	フェ
fo	fo	ふぉ	フォ
hya	hya	ひゃ	ヒャ
hyu	hyu	ひゅ	ヒュ
hyo	hyo	ひょ	ヒョ
mya	mya	みゃ	ミャ
myu	myu	みゅ	ミュ
myo	myo	みょ	ミョ
rya	rya	りゃ	リャ
ryu	ryu	りゅ	リュ
ryo	ryo	りょ	リョ

a)-4 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
gya	gya	ぎゃ	ギャ
gyu	gyu	ぎゅ	ギュ
gyo	gyo	ぎょ	ギョ
gwa	gwa	ぐわ	グワ
<u>ja</u>	<u>zya</u>	じゃ	ジャ
<u>ju</u>	<u>zyu</u>	じゅ	ジュ
<u>je</u>	<u>zye</u>	じえ	ジェ
<u>jo</u>	<u>zyo</u>	じょ	ジョ
<u>ja</u>	<u>dya</u>	ぢゃ	ヂャ
<u>ju</u>	<u>dyu</u>	ぢゅ	ヂュ
<u>jo</u>	<u>dyo</u>	ぢょ	ヂョ
bya	bya	びゃ	ビャ
byu	byu	びゅ	ビュ
byo	byo	びょ	ビョ
pya	pya	ぴゃ	ピャ
pyu	pyu	ぴゅ	ピュ
pyo	pyo	ぴょ	ピョ
va	va	-	ヴァ
vi	vi	-	ヴィ
vu	vu	-	ヴ
ve	ve	-	ヴェ
vo	vo	-	ヴォ
dya	dya	でい	デイ
tya	tya	てい	テイ

a)-5 Comparison of the Hepburn system and Japanese system

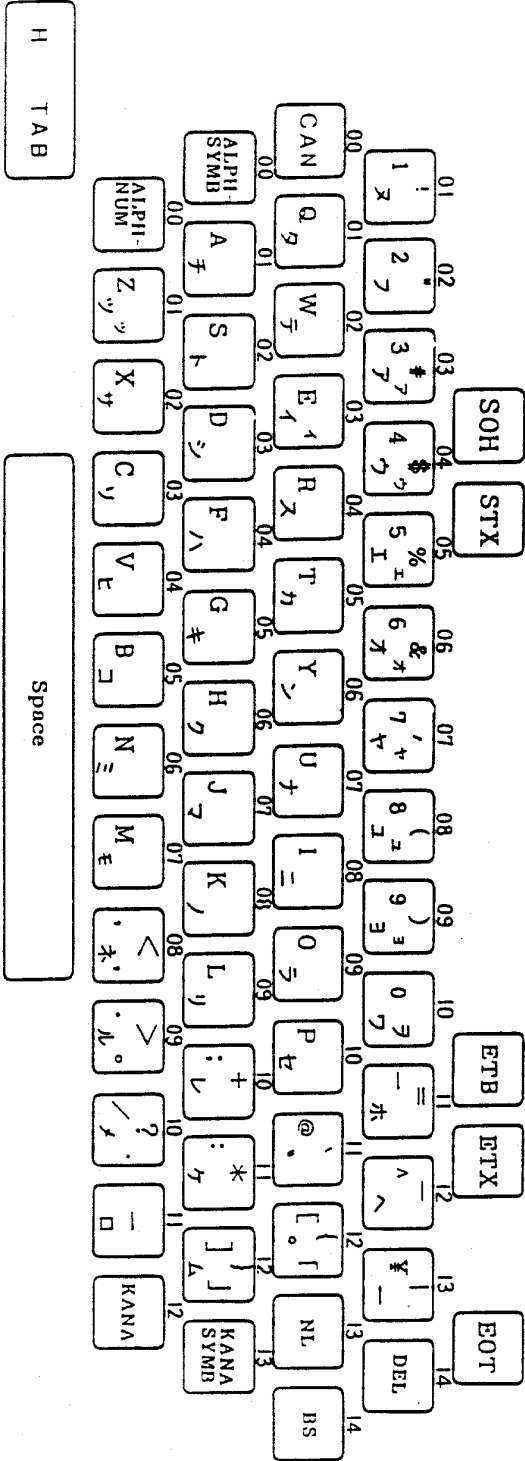


Figure 4.7: Katakana Keyboard, Page 1

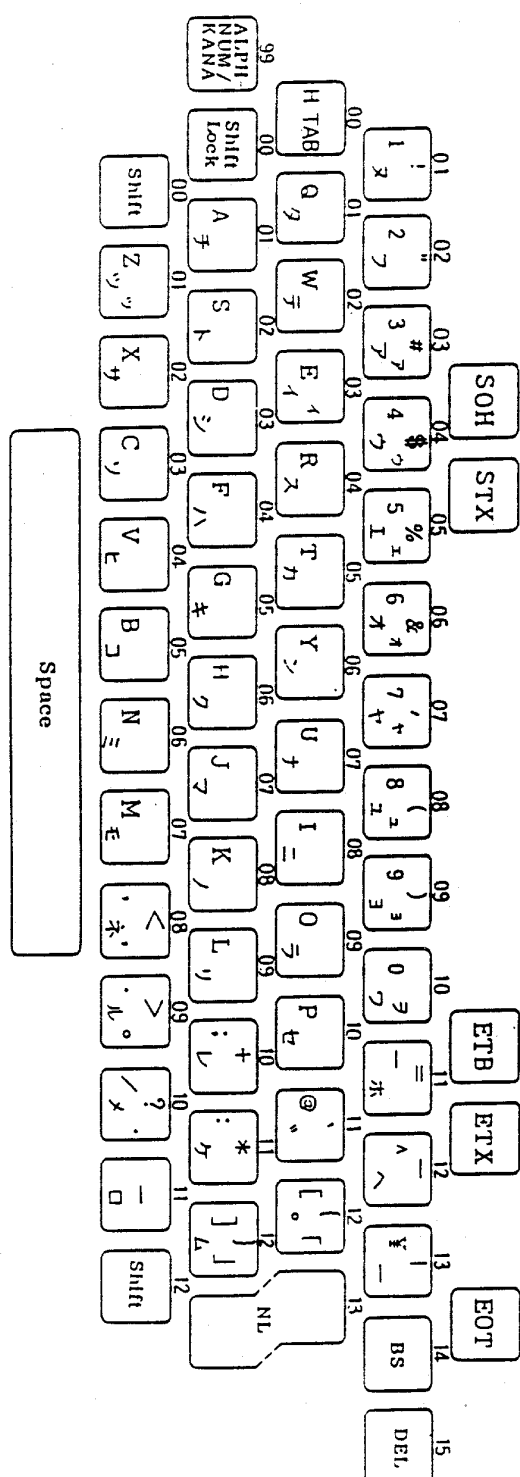
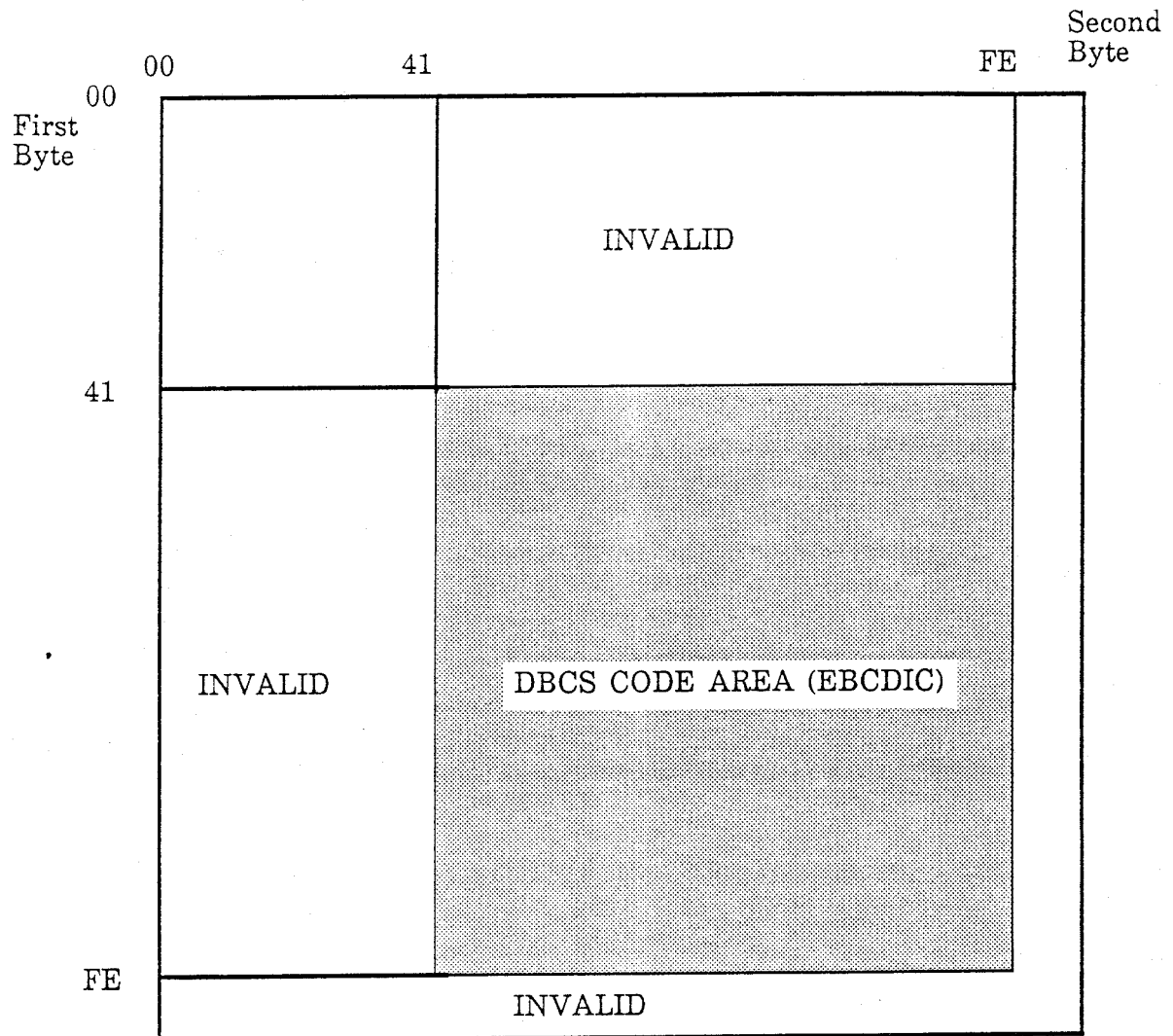


Figure 4.8: Katakana Keyboard, Page 2

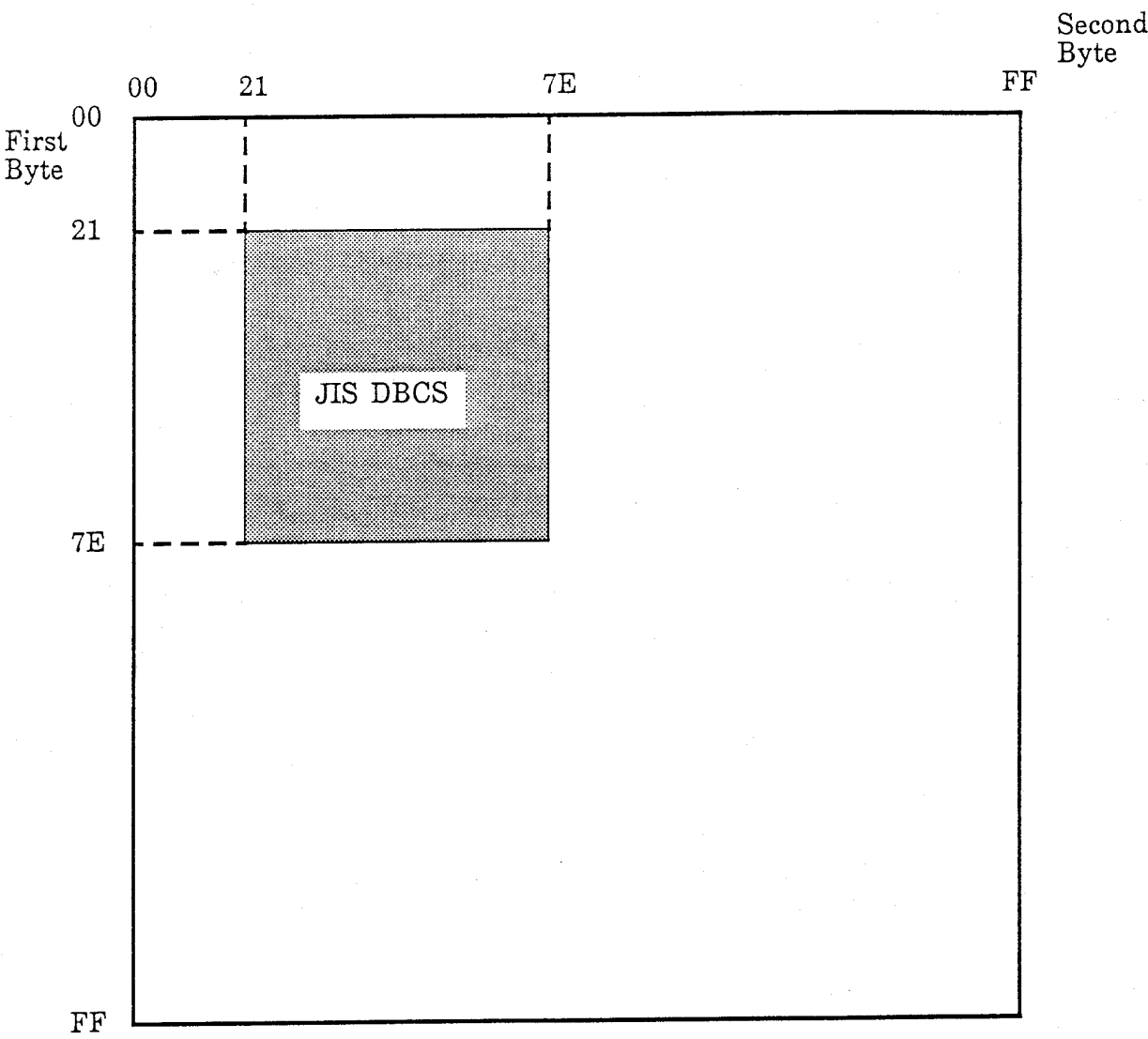
Column ----- Row	00	01	02	03	04	05	06	07
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[k	{
C	FF	FS	,	<	L	¥	l	
D	CR	GS	-	=	M]	m	}
E	SO	RS	.	>	N	^	n	-
F	SI	US	/	?	O	—	o	DEL

Column ----- Row	08	09	0A	0B	0C	0D	0E	0F
0	■	■	■	-	タ	ミ	■	■
1	■	■	。	ア	チ	ム	■	■
2	■	■	「	イ	ツ	メ	■	■
3	■	■	」	ウ	テ	モ	■	■
4	■	■	、	エ	ト	ヤ	■	■
5	■	■	・	オ	ナ	ユ	■	■
6	■	■	ヲ	カ	ニ	ヨ	■	■
7	■	■	ア	キ	ヌ	ラ	■	■
8	■	■	イ	ク	ネ	リ	■	■
9	■	■	ウ	ケ	ノ	ル	■	■
A	■	■	エ	コ	ハ	レ	■	■
B	■	■	オ	サ	ヒ	ロ	■	■
C	■	■	ヤ	シ	フ	ワ	■	■
D	■	■	ユ	ス	ヘ	ン	■	■
E	■	■	ヨ	セ	ホ	〃	■	■
F	■	■	ツ	ソ	マ	。	■	■

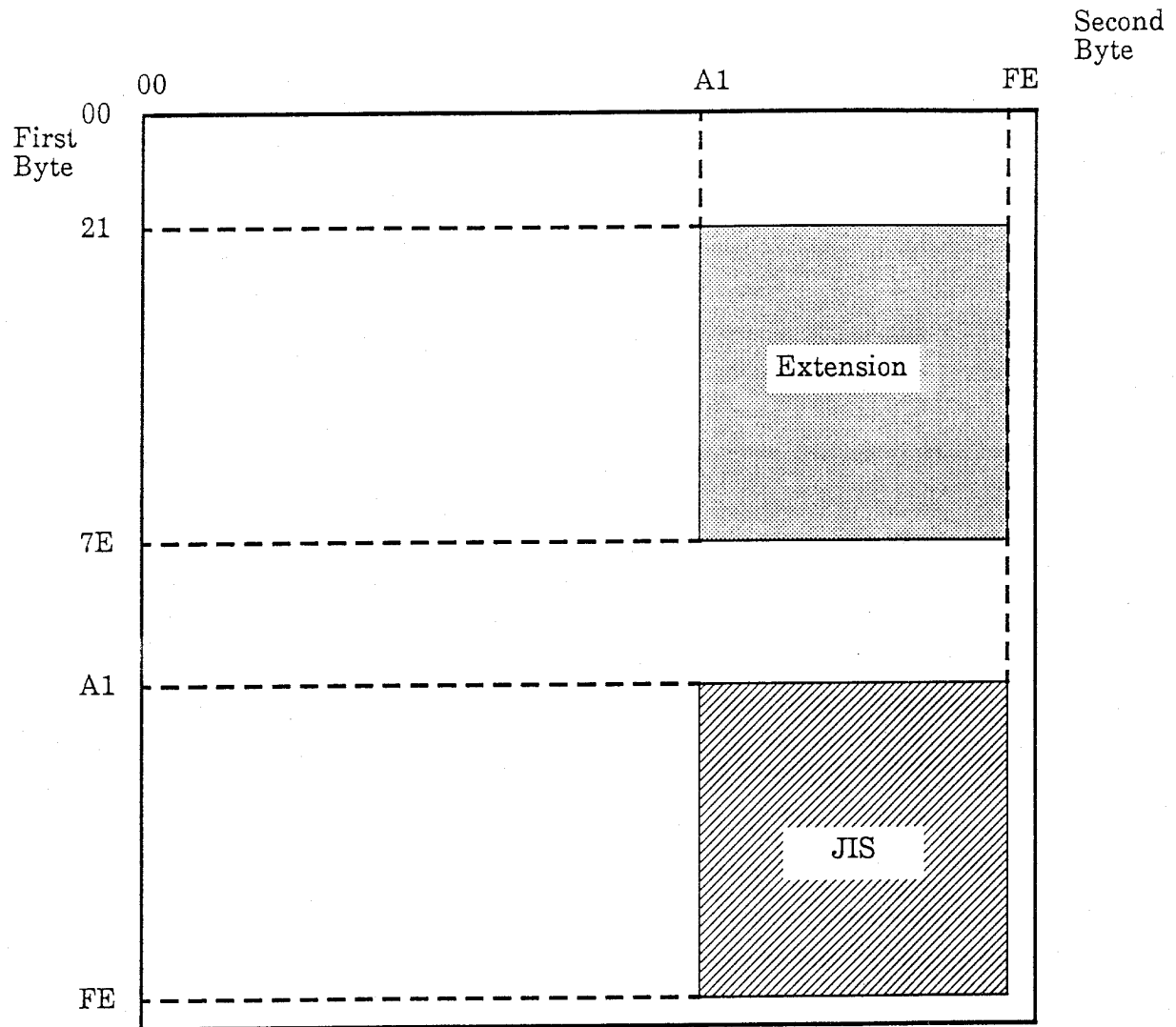
Figure 4.11: The Joyo Kanji character table



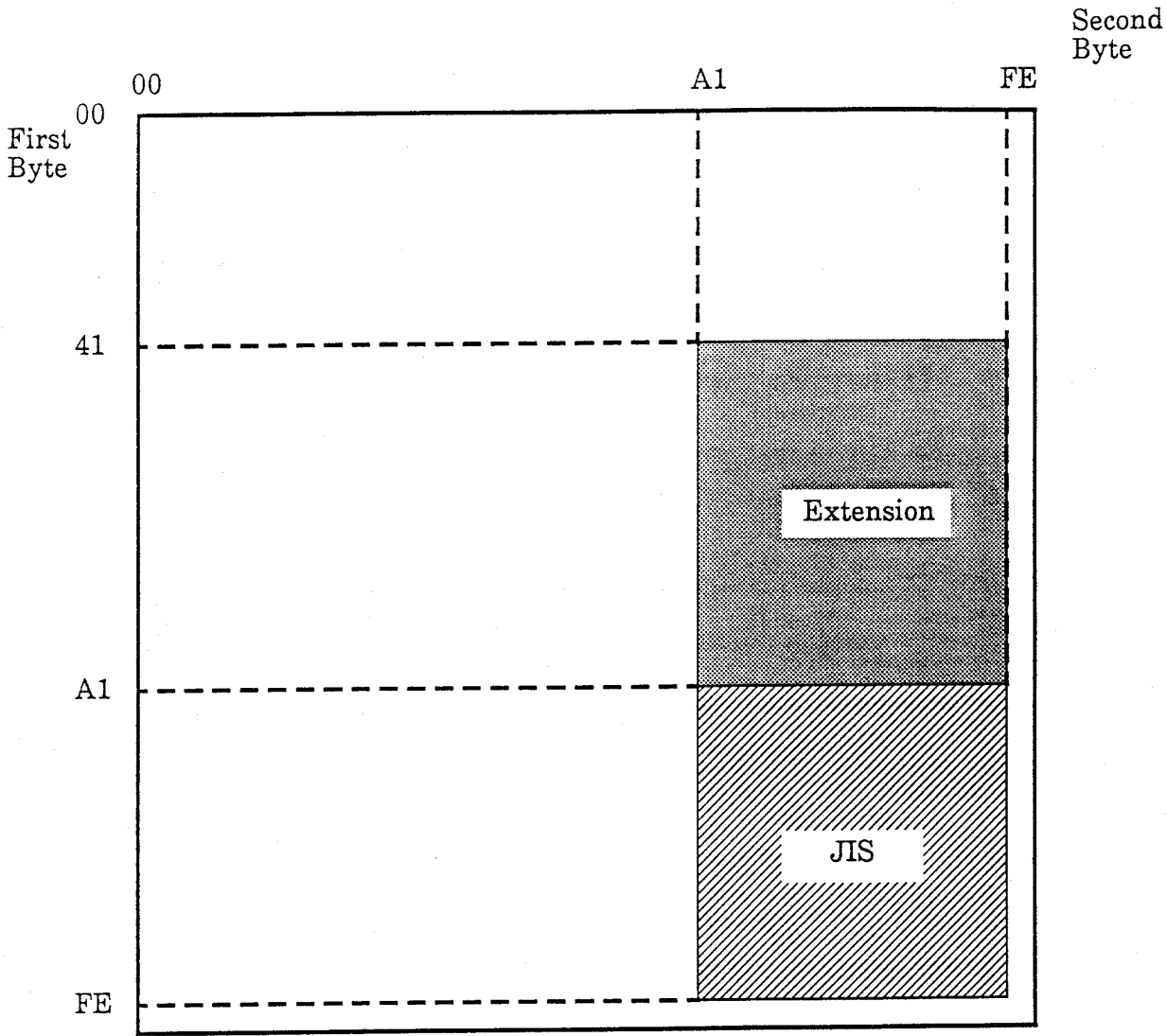
IBM EBCDIC DBCS CODE TABLE



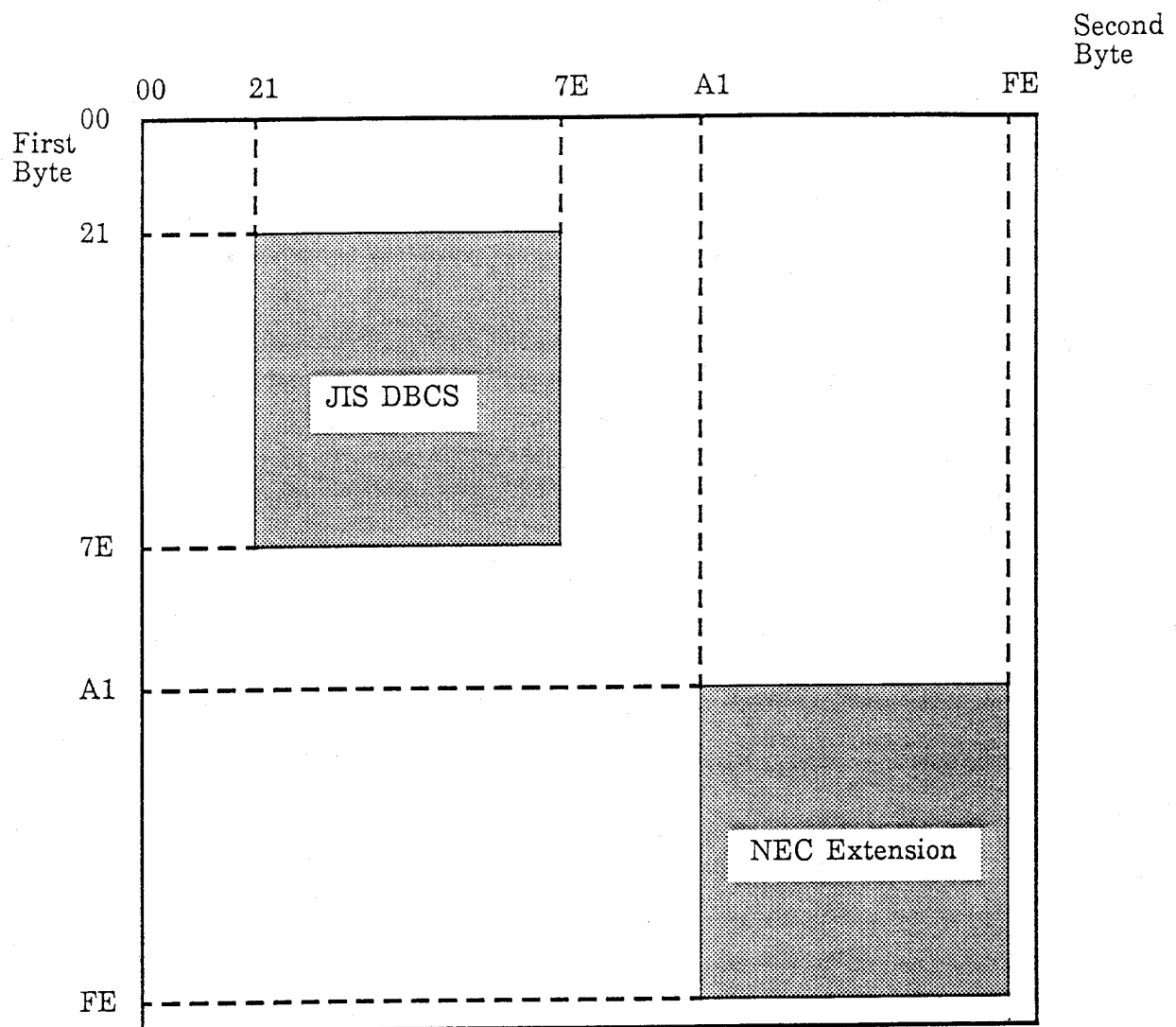
JIS DBCS Code Table



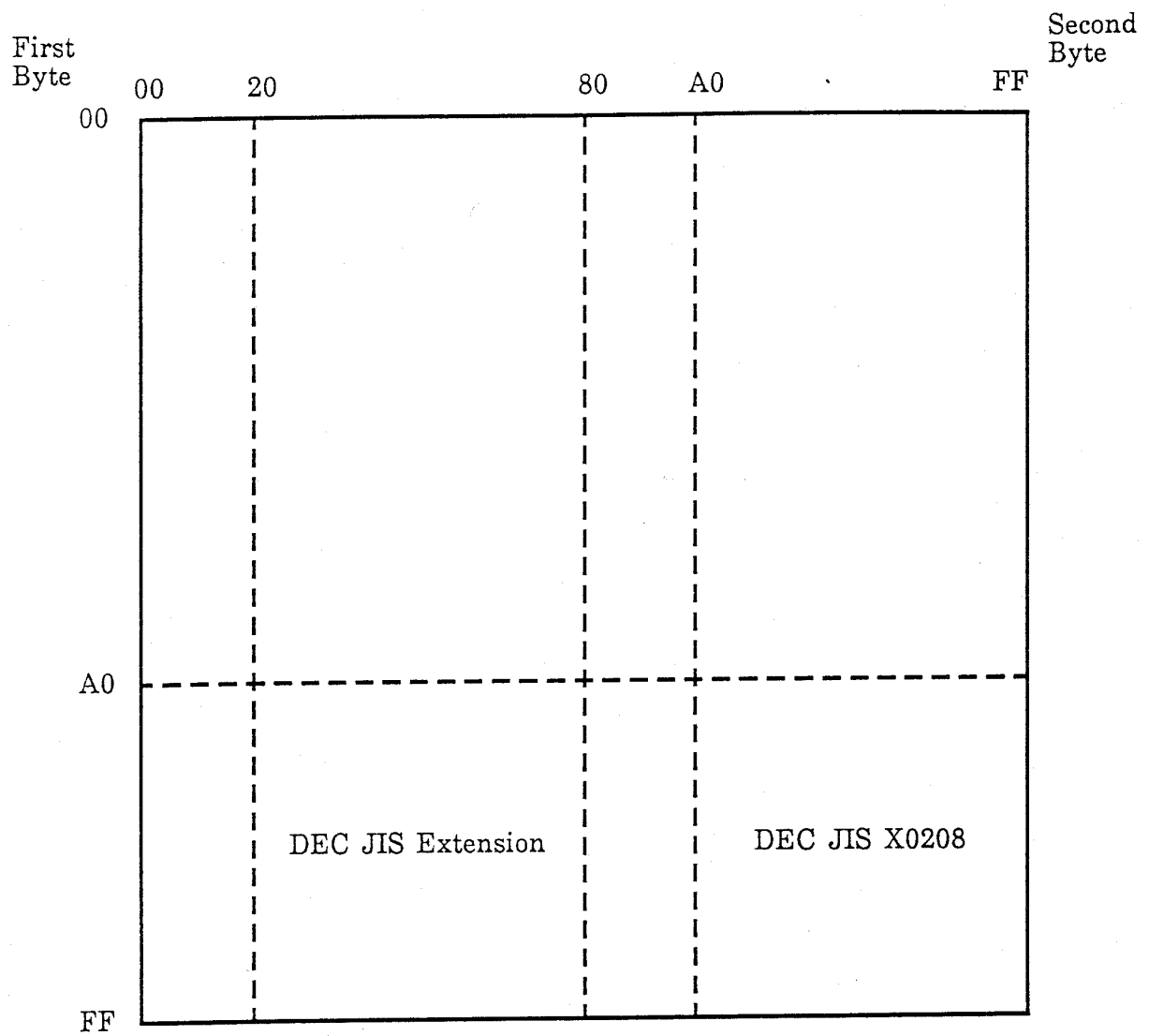
LETS-J JIS DBCS Code Table



JBIS, Fujitsu & Hitachi JIS DBCS Code Table



NEC JIS DBCS Code Table



DEC DBCS Code Table

First Byte	Second Byte	
	00	40 7E 80 FC
00		
81		
9F		
E0		
FC		

Shift JIS DBCS CODE TABLE

Figure 4.18:

4.2.4 DBCS Problems

If a system uses SBCS and DBCS character sets you have to be aware of some arising problems. If you use only SBCS characters you could work like in the "normal" ASCII environment (performing byte oriented operations). It is the same if you work only with DBCS characters. The trouble begins if you start to use mixed strings containing SBCS and DBCS characters.

If your system (operating system, programming language, application) is able to handle both types of characters you have to be aware that the Japanese DBCS characters (Zenkaku) have the double width of a standard ASCII SBCS or JIS SBCS (Hankaku). If you, e.g., design screen masks or reports you have to consider that the Japanese user could enter both SBCS and DBCS. *In the following example's S refers to a SBCS character and DD refers to a DBCS character.*

First an example of a SBCS character string :

S	S	S	S	S
---	---	---	---	---

(length 5 Bytes)

Now a five characters long DBCS string :

DD	DD	DD	DD	DD
----	----	----	----	----

(length 10 Bytes)

Here you will see a five character long mixed string :

S	S	DD	S	DD
---	---	----	---	----

(length 7 Bytes)

As mentioned above the length of all strings is five characters, but the actual length of the displayed string is different. If you design a screen mask or a report you have to take this in consideration and make your design so that it is able to handle all three types of strings (SBCS, DBCS and mixed strings). A programming language or application has to be able to handle strings with SBCS and DBCS. For example, the database system Oracle, uses no special data type for DBCS characters. The Japanese version of Oracle is able to handle DBCS characters. If you create a table with a 20 character long data field, this field could accept SBCS and DBCS characters as input. You could enter up to 20 SBCS characters or up to 10 DBCS characters (2 byte for each character). If you enter a mixed SBCS/DBCS string the maximum length can not exceed the length of 20 byte (a mixed string could contain, e.g., 10 SBCS and 5 DBCS characters which represent a 20 byte long string). This strategy has the

advantage that you not have to change the definition of a table. In addition the screen mask and report layout have not to be redesigned. It gets even more difficult if your system uses SI/SO (KI/KO) sequences. The same example with SI/SO sequence would look like this :

The first example with a five character SBCS string :

S	S	S	S	S
---	---	---	---	---

 (length 5 Bytes)

Now again a five characters long DBCS string with SI/SO sequence :

$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	DD	DD	DD	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$
--	----	----	----	----	----	--

 (length 12 Bytes)

A five character long mixed string with SI/SO sequence would look like :

S	S	$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$	S	$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$
---	---	--	----	--	---	--	----	--

 (length 11 Bytes)

You will recognize that the strings become longer through the use of the SI/SO sequence. The SI/SO sequence will not be displayed on the screen, but it makes the string, at least two bytes (depending on the implementation of the vendor), longer. In addition the use of a SI/SO sequence makes it more difficult to handle, e.g., string operations. These operations are operations like cursor movement, backspace, wrapping of strings, deletion or insertion and windowing.

First I will explain the handling of strings with a SI/SO sequence and later I will give you some examples about the handling of mixed strings without SI/SO sequence. The difference between the handling is that you have to be aware of the SI/SO sequence. It is easier to handle strings without SI/SO sequence but also here you have to be aware of the difficulties in the handling of mixed strings.

Handling of mixed strings with SI/SO sequence

If you press a cursor key to move forward or backward through a string the system has to increase or decrease the pointer of the actual cursor position. This would look like this :

S	S	<u>S</u>	S	S
---	---	----------	---	---

cursor right (\rightarrow) pressed once

S	S	S	<u>S</u>	S
---	---	---	----------	---

cursor left (\leftarrow) pressed once

S	S	<u>S</u>	S	S
---	---	----------	---	---

This works quite fine in a SBCS environment which works byte oriented. If we now have a look at the DBCS string, we will see that we no longer can use the byte oriented approach. In order to handle DBCS correctly we have to use a character oriented approach :

byte processing

<u>S</u>	I	DD	DD	<u>DD</u>	DD	DD	<u>S</u>	I
----------	---	----	----	-----------	----	----	----------	---

cursor right (\rightarrow) pressed once

<u>S</u>	I	DD	DD	<u>DD</u>	DD	DD	<u>S</u>	I
----------	---	----	----	-----------	----	----	----------	---

cursor left (\leftarrow) pressed once

<u>S</u>	I	DD	DD	<u>DD</u>	DD	DD	<u>S</u>	I
----------	---	----	----	-----------	----	----	----------	---

character processing

<u>S</u>	I	DD	DD	<u>DD</u>	DD	DD	<u>S</u>	I
----------	---	----	----	-----------	----	----	----------	---

cursor right (\rightarrow) pressed once

<u>S</u>	I	DD	DD	DD	<u>DD</u>	DD	<u>S</u>	I
----------	---	----	----	----	-----------	----	----------	---

cursor left (\leftarrow) pressed once

<u>S</u>	I	DD	DD	<u>DD</u>	DD	DD	<u>S</u>	I
----------	---	----	----	-----------	----	----	----------	---

In the byte oriented approach the cursor does not move to the next character if you press it once. It moves from the first byte of the DBCS character to the second byte. Only in the character oriented version the cursor moves to the next character.

This gets even more complicated if you have to handle mixed strings which contain SBCS and DBCS characters :

byte processing

S	<u>S</u>	<u>S</u>	I	DD	<u>S</u>	O	S	<u>S</u>	I	DD	<u>S</u>	O
---	----------	----------	---	----	----------	---	---	----------	---	----	----------	---

cursor right (\rightarrow) pressed once

S	S	<u>S</u>	I	DD	<u>S</u>	O	S	<u>S</u>	I	DD	<u>S</u>	O
---	---	----------	---	----	----------	---	---	----------	---	----	----------	---

character processing

S	<u>S</u>	<u>S</u>	I	DD	<u>S</u>	O	S	<u>S</u>	I	DD	<u>S</u>	O
---	----------	----------	---	----	----------	---	---	----------	---	----	----------	---

cursor right (\rightarrow) pressed once

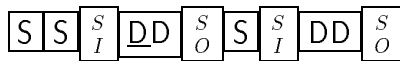
S	S	<u>S</u>	I	<u>DD</u>	<u>S</u>	O	S	<u>S</u>	I	DD	<u>S</u>	O
---	---	----------	---	-----------	----------	---	---	----------	---	----	----------	---

In the byte oriented example the cursor moves to the next byte, the SI sequence, and not to the next character. In the character oriented system the routine which moves the cursor has to detect the start of a DBCS character by recognizing the SI sequence. Then the routine has to set the cursor to the first byte of the DBCS character.

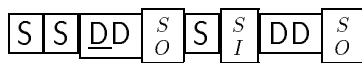
Similar problems occur if we press the backspace key to delete a character. In the next example you will see that in the byte oriented version the SI sequence gets lost and

leaves us with a garbage string. This string will no longer displayed correct because the SI sequence is missing and so the system is not able to recognize the start of the DBCS characters. Depending on the implementation the system will try to display the bytes of the DBCS character as SBCS characters (e.g., as Katakana or ASCII character depending on the code).

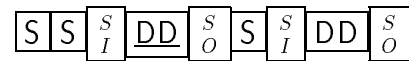
byte processing



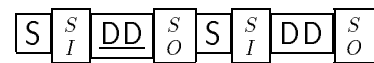
Backspace (\Leftarrow) pressed once



character processing



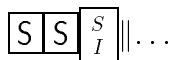
Backspace (\Leftarrow) pressed once



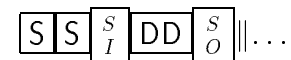
The same type of problem will occur if you truncate (e.g., the third character), exchange a character or insert a character (or string) in the mixed string.

Truncation after the third character

byte processing

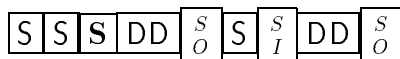


character processing

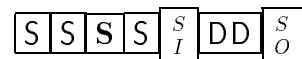


Exchange of a SBCS character at the third position

byte processing



character processing

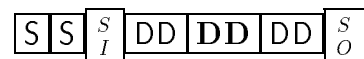


Exchange of DBCS character at the fourth position

byte processing

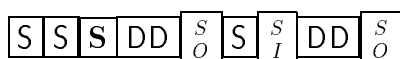


character processing

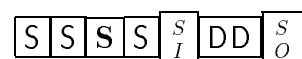


Insertion of a SBCS character at the third position

byte processing



character processing



Insertion of a DBCS character at the second position



As you may recognize the routines have to take care not only of the DBCS characters but also of the SI/SO sequence.

This breed of problem will also affect wrapping a string, searching a character in a string, upper and lower casing or working with windows on a screen. In the case of an environment which works with SI/SO sequences the handling of mixed strings is more complicated than in a system which works without SI/SO sequences. Nevertheless it is necessary to be aware of the mixed string problems in a, e.g., Shift-JIS environment.

Handling of mixed strings without SI/SO sequences

Again I will start with the cursor movement. In picture A in figure 4.19 on page 93 (all examples [24]) you will see the cursor movement in a byte oriented environment. All characters in the picture are DBCS characters. There is no SI/SO sequence required because the MSB of this type of DBCS characters is always set to one. If we now press to cursor right key the cursor will move one byte to the right. From the first byte of a DBCS character (in line 1) to the second byte of the DBCS character (in line 2) and so on. In picture B (in figure 4.19) you will see how the cursor will move if the system works character oriented instead of byte oriented. The cursor jumps from one character to the next DBCS character. If we had a mixed string the system could distinguish between SBCS and DBCS characters by looking at the MSB. If the MSB is zero the pointer for the cursor position has to be moved one byte. If the MSB is one the system has to process a DBCS character and has to move the pointer for the cursor position two bytes instead of one byte.

As mentioned above you have to check which type of character you will process, e.g., if you press the backspace key. If have a DBCS character string and the system works in byte oriented mode the press of the backspace key will delete just one byte of the string. This will leave us with a corrupted string (see picture A in figure 4.20 on page 94). As you see some of the characters will change because the byte pairs where change through the byte oriented backspace operation. On the opposite this will not happen if you use a character oriented operation. In this case the system would correctly delete one DBCS character, i.e., one byte pair.

On page 95 in figure 4.21 you will see in picture A the byte oriented replace operation. After typing the (SBCS) character " H " the system will replace just one byte and so the mixed string will be corrupted. The Kanji character CBDC_{H_{ex}} (Hon or Moto) will partly be replaced by the " H " (48_{H_{ex}}). The character " D " (44_{H_{ex}}) will form with the second byte from the DBCS character a new (illegal) DBCS character. If the system works in the character oriented mode the replace operation will replace the Hon (or Moto) Kanji character with either one byte and deletes the second byte of the DBCS character or it replaces the first byte with the code for " H " (48_{H_{ex}}) and the second byte of the DBCS with the SBCS code for space (20_{H_{ex}}).

The handling of mixed character string's effects all string handling operations of a computer system. For example if we use the operations for upper or lower casing the system has to distinguish between byte processing and character processing. In picture A in figure 4.22 (page 96) you will see the upper casing (No. 1) and lower casing (No. 2) operation performed in the byte oriented mode. This will corrupt the string. If we use the character oriented operations the string will be processed correctly (see picture B, No.1 & 2 in figure 4.22). A group of similar problems are the string wrapping and substring operations. On page 97 in figure 4.23 you will see what happens when the system has to perform a line-wrap (e.g., at the end of a line). If the space which is left at the line is 18 bytes and the string is longer then this space (here 21 bytes) the system has to wrap the line and put some of the characters in the next line. Is this operation performed byte oriented the system would wrap incorrectly and would split a DBCS character (picture A in figure 4.23). This would cause that the first character in the next line is incorrect. If the system works character oriented it would use only 17 of the 18 bytes and would split the string correct (picture B in figure 4.23). The same problem appears if we try to move a longer string into a shorter string data field. In the example in figure 4.24 on page 98 the original string is 21 Bytes long. If a byte oriented routine moves this string into a 10 byte long data field the system would cut the string after 10 bytes and would leave us with a corrupted DBCS character (picture A). The character oriented operation instead would copy only 9 byte because the next character after the C is a DBCS character which would not fit in the remaining space.

As last example for the splitting of DBCS characters I will show you a windowing operation performed in byte oriented and character oriented mode. In picture A on

page 99 (figure 4.25) you will see that the DBCS character is " split ". In a GUI this would not cause any problems but if you work with a character oriented program (in text mode) you have to be careful where you place the border of your window. You can not split SBCS character (because they are only one byte long and placing a border above it will always hit the whole character) but if you place the border of a window on the second byte of a DBCS character this character will be corrupted. To perform the windowing operation correctly in a DBCS environment you (or your program) have to watch where it places the border of a window (see picture B in figure 4.25).

Another function which could give us funny results is the search string function. If you perform a byte oriented search operation on the source string in figure 4.26 (page 100) with the parameter `C7A5Hex` the function will return 2 positions in the source string (picture A). One of this positions is the second byte of the third DBCS character and the first byte of the fourth DBCS character. The correct result is that only the last DBCS character matches the search string (picture B).

If you use DBCS characters with or without SI/SO sequence you always have to be aware of the arising problems in handling these characters. If you use the byte oriented operations of an English or American system you will get some strange results. Only if you use the correct processing method (character oriented) your system will give you the results which you expect.

日本ディジタルイクイップメント株
日本ディジタルイクイップメント株
日本ディジタルイクイップメント株
日本ディジタルイクイップメント株

A) Byte processing

日本ディジタルイクイップメント株
日本ディジタルイクイップメント株
日本ディジタルイクイップメント株
日本ディジタルイクイップメント株

B) Character processing

Figure 4.19: Cursor Movement

日 本 デ ィ ジ タ ル 株 式 会 社

C6FC CBDC A5C7 A5A3 A5B8 A5BF A5EB B3F4 BCB0 B2F1 BCD2

日 本 デ ィ ジ タ コ  芦 饅

C6FC CBDC A5C7 A5A3 A5B8 A5BF A5B3 F4BC B0B2 F1BC D2

A) Byte processing

日 本 デ ィ ジ タ ル 株 式 会 社


C6FC CBDC A5C7 A5A3 A5B8 A5BF A5EB B3F4 BCB0 B2F1 BCD2

日 本 デ ィ ジ タ 株 式 会 社

C6FC CBDC A5C7 A5A3 A5B8 A5BF B3F4 BCB0 B2F1 BCD2

B) Character processing

Figure 4.20: Backspace operation

日	本	D	E	C	株	式	会	社
C6FC	CBDC	44	45	43	B3F4	BCB0	B2F1	BCD2
日	H		E	C	株	式	会	社
C6FC	48	DC44	45	43	B3F4	BCB0	B2F1	BCD2

A) Byte processing

日	本	D	E	C	株	式	会	社
C6FC	CBDC	44	45	43	B3F4	BCB0	B2F1	BCD2
日	H	D	E	C	株	式	会	社
C6FC	48 (20)	44	45	43	B3F4	BCB0	B2F1	BCD2

B) Character processing

Figure 4.21: Replace function

日 本 デ ッ ク 株 式 会 社 dEc

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 64 45 63

Text before processing

1) 頓 本 デ ッ ク 郭 式 英 社 DEC

C6DC CBDC A5C7 A5C3 A5AF B3D4 BCB0 B2D1 BCD2 44 45 43

2) 脣 潜 ヨ ヤ ク 株 式 会 酒 dec

E6FC EBFC A5E7 A5E3 A5AF B3F4 BCB0 B2F1 BCF2 44 65 63

A) Byte processing

1) 日 本 デ ッ ク 株 式 会 社 D E C

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 44 45 43

2) 日 本 デ ッ ク 株 式 会 社 dec

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 64 65 63

B) Character processing

Figure 4.22: Upper and Lower casing

日 本 デ D E C ッ ク 株 式 会 社

C6FC CBDC A5C7 44 45 43 A5C3 A5AF B3F4 BCB0 B2F1 BCD2

Text before processing

日 本 デ D E C

C6FC CBDC A5C7 44 45 43

A5

A) Byte processing

日 本 デ D E C

C6FC CBDC A5C7 44 45 43

B) Character processing

Figure 4.23: Word wrap

日 本 デ ッ ク D E C 株 式 会 社
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 B2F1 BCD2 (21 BYTE)

Text before processing

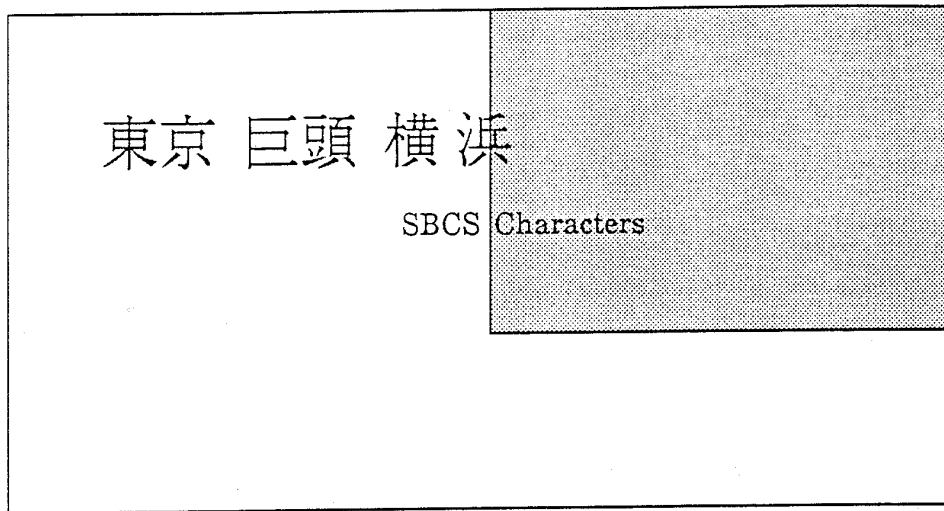
日 本 デ ッ ク D E C 株 式
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 B2 (18 Byte)
 僅
 F1BC D2

A) Byte processing

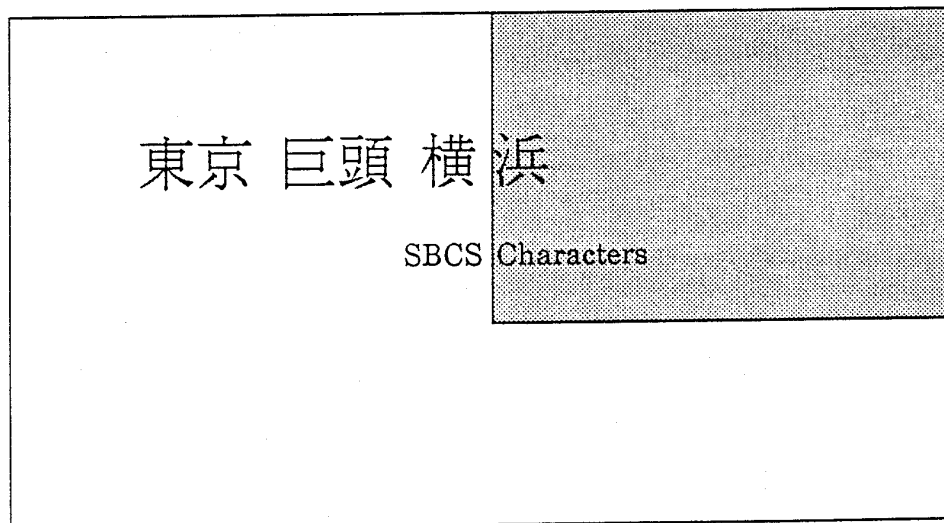
日 本 デ ッ ク D E C 株 式
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 (17 BYTE)
 会 社
 B2F1 BCD2

B) Character processing

Figure 4.24: Line truncation



A) DBCS Character split by Window



B) Correct DBCS Character windowing

Figure 4.25: DBCS windowing

日本デック株妊
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

Source String

Search String : 妊 (C7A5)

日本デック株妊
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

A) Byte processing

日本デック株妊
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

B) Character processing

Figure 4.26: Search string function

4.3 Number & Currency convention

In the Japanese business world many computer programs rely on numbers, monetary values and the proper handling of this kind of information are very important.

4.3.1 Number representation

The presentation of numbers varies only in the use of different separators for a group of three digits. Table 4.4 (on page 101, [1]) shows you an example of different formats for the representation of numbers. As you see in the table there are only

Country	Positive Numbers	Negative Numbers
Germany	12.345,67	-12.345,67
Australia	12,345.67	-12,345.67
Arabic	123.45,67	123.45,67-
France	12 345,67	-12 345,67
South Africa	12 345.67	(12 345.67)
Japan	12,345.67	-12,345.67

Table 4.4: Number Notations

few variations in the representation of a number. As separators are used the period, comma and space. The minus is mainly put in front of a negative number.

In addition to the western way of writing numbers the Japanese use their traditional Kansuji characters (see figure 4.27 on page 103). In this way of writing the Japanese use special Kanji characters to express 10 (Ju), 100 (Hyaku), 1,000 (Sen, do not mix up with Sen, each Sen has a different Kanji character), 10,000 (Man), 100,000,000 (Oku) and 1,000,000,000 (Cho).

For expressing fractions, ordinal numbers and percentage the Japanese use the western way and their traditional way. A fraction $\frac{N}{M}$ is written in Japanese with the Bun No as the fraction stroke (see picture b in figure 4.28 on page 109). Ordinal numbers are written with the Bam-me as sign (see picture c in figure 4.28). For expressing a percentage the Japanese use Wari ($\frac{1}{10}$), Bu ($\frac{1}{100}$) and Rin ($\frac{1}{1000}$). If you want to

express 56.7 % you could write it like in picture d in figure 4.28. All of these numbers will be written with western style numbers or the traditional Kansuji characters (see figure 4.27 on page 103).

4.3.2 Rounding of Numbers

For business purposes the Japanese use the normal way of rounding numbers, i.e., from XXX.0 to XXX.4 the system must round to XXX. In the case of that the number is between XXX.5 and XXX.9 the system has to round to XX(X+1). For example : 123.454 rounds to 123.45 and 123.457 rounds to 123.46 ([1]). In some business areas it is common to round to the next 1,000 or 10,000 Yen ([11]).

4.3.3 Currency

The Japanese currency is called Yen (or En). One Yen is 100 Sen. Today the Sen is mainly used in the banking area. Yen is represented as a one byte character (SBCS) and En is a two byte (DBCS) character. In everyday life the Japanese use only the Yen. The effect is that there are no currency decimal positions used in Japan (One Yen is the smallest coin and 10,000 Yen the biggest bill). The international representation for the Japanese Yen is, regarding to ISO 4217 (Codes for the representation of currency and funds), the JPY (see table D in figure 4.32 on page 114) symbol. The small difference in the use of Yen and En is that the Yen is placed in front of the amount and the En is placed behind the amount of money. For an example for the writing of positive and negative amounts of money see table D in figure 4.32. The representation of positive and negative values follows the standard for writing numbers mentioned above.

For a data field which represent a monetary value you should reserve a currency field length among 12 and 15 digits ([1], [11]). In this field you have to put the Yen (SBCS) or En (DBCS) symbol and a comma every three digit.

Number	Name	Name in Kanji	Name in Kansuji
0	Zero, Rei	零	〇
1	Ichi	壹	一
2	Ni	貳	二
3	San	参	三
4	Yon, Shi	四	四
5	Go	五	五
6	Roku	六	六
7	Nana, Shichi	七	七
8	Hachi	八	八
9	Kyu	九	九
10	Ju	十	十
100	Hyaku	百	百
1,000	Sen	千	千
10,000	Man	万	万
100,000,000	Oku	億	億
1,000,000,000,000	Cho	兆	兆

Table of Kanji and Kansuji Characters of Japanese Numbers

For example, 1992 is written 一千九百九十二

Figure 4.27:

4.4 Date & Time convention

In this section I would like to introduce the reader to the different date and time formats which are used in Japan. Furthermore I will explain the different calendar systems that are used in Japan.

4.4.1 Date formats

In this section I would like to introduce the reader to the differences in the representation of a date. In some cultures (e.g., Taiwan, Thailand, Arab countries, Israel, Japan) is not only the Gregorian calendar system used. Besides this calendar these countries use one or more other calendar systems based on their culture or religion (see, e.g., [1], 5-2).

In Japan there are two types of calendar systems commonly used :

- The Gregorian calendar system The first system is our commonly used Gregorian calendar system. With 365 days a year, 12 months with 30, 31 or 28 (in a leap year 29) days. The difference between the date convention in, e.g., Germany, is that the Japanese got a different style of writing a date, here you will see some examples² for different styles of date representation (see table 4.5).

Country	Representation	Example
Germany	DD . MM . YYYY	10.02.1966
Australia	DD / MM / YYYY	26/07/1991
USA	MM - DD - YYYY	03-20-1992
Japan	YYYY - MM - DD	1992-03-26

Table 4.5: Gergorian Date represantation

The system, which is used in Japan is the year - month - day representation. This way got the advantage that, if you want to bring something in chronologic (or reverse chronologic) order, it is very easy to do.

²DD refers to the day, MM to the month and YYYY refers to the year

- Gengou calendar system (see picture a in figure 4.29 on page 111)

The second heavily used system (e.g., from the Japanese government, the most governmental forms expect this date format) is based on the reign of the Japanese tenno (emperor). After the death of the emperor a commission decides the name for the next era. Then this new era starts from the year one. Shouwa was the name of the last era, which ended 1989, in the 64th year of this era, with the death of emperor Hirohito. So that today, in the year 1992, is the 4th year of the actual tennos reign. His name is Akihito and the actual era is called the Heisei era. (*Historic note : in this date format the Japanese use the terms Seireki and Kigenzen for dates before the Meiji era, see figure 4.28 on page 109*). This Gengou (sometimes also called Nengou) type of calendar system gives the computer systems and software some problems on the way :

- The system should ” *know* ” if there is a coronation of a new emperor, so that the name of the era could be changed. Also the counter for the year has to be set to one. An operator has to enter this information when a change of the emperor occurs. Also the system should provide a function to transfer from the Gregorian calendar system to the Gengou calendar system and reverse.
- The printed or displayed presentation of the Gengou calendar system is possible in several ways :
 - * A short, modern form is EY - MM - DD, like 4 - 4 - 17, which is e.g., the 4th year of the era, April, the 17th. EY represents the year of emperors reign, the era year. This form is the easy everyday used form for, e.g., tickets, advertising. Besides that form there is a slightly different form used. (see picture b-1 in figure 4.29 on page 111) In front of the EY (emperor’s year) is printed (or displayed) the first letter of the era name, like H 4 - 4 - 17. So that it is easier to see which era is meant. In the first example it does not have to be a date in the Heisei era, it also could be a date in, e.g., the Meiji era.
 - * An other, often used form, is the EY KY MM KM DD KD form as date representation. With the era year and the Kanji Nen (KY, year), then the month and the Kanji Tsuki (KM, month) and at last the day with the Kanji Hi (KD, day). This date representation would look for

the 16th of April, 4th year of the era, like in picture b-2 in figure 4.29 on page 111.

Sometimes, e.g., cash registers use a similar form. Instead of the EY (emperor year) you will see the Gregorian calendar system year. This mixed form is sometimes used, because you do not have to change the sign for the era. It still uses the Kanji characters for year, month and day.

- * The third form is an extended version of the second form. For this form the system has to add the name of the era in Kanji characters in front of the second form. In the year 4 of the Heisei era (1992) , at April the 16th, this form would look like in picture c in figure 4.29 on page 111.
- * Besides this two advanced ways it is possible to use the Kansuji for the month (see figure 4.27 on page 103 or the Kanji characters for the Japanese name of the month. You will find a table with this representation of the months in figure 4.30 on page 112). Today this form of date representation is a little bit unfashionable, but it could be used under some circumstances (see picture d in figure 4.29 on page 111). Nevertheless it is possible to write the whole date in Japanese Kanji characters (see picture e in figure 4.29 on page 111) in this case you could use the kansuji (see figure 4.27 on page 103) or the traditional Japanese name of the day (see figure 4.31 on page 113).
- Except for the first forms your system has to be able to display or print Kanji characters, if you want to use the advanced forms of date representation in the Gengou calendar system.

Besides the era year and the additional Kanji characters does this calendar version work like the Gregorian calendar system (12 months, a 30, 31 or 28 (29 in a leap year) days, see [1], 5-4). . In table 4.6 I will give a short overview about the last emperor eras in Japan.

The Kanjis for these eras would look like picture a in figure 4.28 on page 109.

Following the JIS standard JIS X0301-1977 (replaced this year with JIS X0301-1992) the JSA recommends three different time formats (see [29]) :

1. For EDI purposes the format YYMMDD (920326) or YYYYMMDD (19920326).
2. For the human machine interaction the JSA recommends the form YY-MM-DD (92-03-26) or YYYY-MM-DD (1992-03-26). A second recommended specification is the YY MM DD or YYYY MM DD form.
3. The form, which corresponds to the Japanese calendar, is YY.MM.DD (4.03.26) or *era*YY.MM.DD (H4.03.26, with H = Heisei, S = Showa, T = Taisho and M = Meiji)

4.4.2 Time formats

Like the most other countries in the World the Japanese use for business purposes the 24 hour time system (see [1], 5-5). Again, the difference is based on the representation of the data. But in this case the difference is just a minor difference. The separator, which separates hour, minute, and second is varying. Some countries like e.g., Argentina, Denmark, Italy, use the period as separator (" . "). In the most other countries use the colon (" : ") instead. In table 4.7 (page 110) the time that is represented is 22 hours (10 pm), 42 minutes, 00 seconds and an additional 30 milliseconds, if the milliseconds appear in the used time format. The separator for "fractions of a second" should, referring to the ISO (International Standardization Organization) standards ISO 3307 and ISO 1000, be the same, which is used as decimal separator in the number representation format, for the country (i.d. [1], 5-5).

The represented forms could be :

- one digit for tenths of a second
- two digits for hundreds of a second
- three digits for thousandths of a second

Besides this, in the business world used time format, there is an other Japanese time format. This time format contains, like the Gengou date representation, some special Kanji characters. The characters are called Ji, which is used for the hour, and Fun, which represents the minutes. So that the time 22:42:00 would look like picture a in figure 4.32 on page 114.

Moreover that time format the Japanese also know a 12 hour representation. This twelfth hour format runs actually from 0 to 11. In that case the Kanji Gogo is used as sign for PM (12:00 – 24:00) and the Kanji Gozen is the sign for AM (00:00 – 12:00). In this type of time format 22:42 would look like picture b in figure 4.32 on page 114. At 10:42 in the morning (AM) it would look like picture c in figure 4.32 on page 114. As in the Gengou date representation your system has to be able to display or print Kanji characters if you want to use this time format.

Following the JIS standard JIS X0302-1977 (combined later this year with the new JIS X0301-1992) the JSA recommends two different time formats (see [30]) :

1. For EDI purposes the format hhmmss (224200).
2. For the human machine interaction the JSA recommends the form hh:mm:ss (22:42:00).

In addition to the already introduced date and time formats there are combined date and time formats recommended by the JSA (see [30]) :

1. For EDI purposes the format YYMMDDhhmmss (920326224200) or YYYYMMDDhhmmss (19920326224200).
2. For the human machine interaction the JSA recommends the form YY-MM-DD-hh:mm:ss (92-03-26-22:42:00) or YYYY-MM-DD-hh:mm:ss. The second recommendation is the (YY)YY MM DD hh:mm:ss form.
3. The Japanese calendar based form looks like (*era*)YY.MM.DD,hh:mm:ss (H4.03.26,22:42:00).

As a last information about the time format in Japan I would like to mention that the Japanese do not use daylight saving time. There is no need to set the clock forward or backward in spring and fall.

a)	Era	Kanji	Emperor
	Meiji	明治	Mutsuhito
	Taisho	大正	Yoshihito
	Showa	昭和	Hirohito
	Heisei	平成	Akihito
	Years before Meiji		
	Seireki(A.D.)	西暦	Anno Domini
	Kigenzen(B.C.)	紀元前	Before Christ
b)	Fractions		
	$\frac{N}{M}$ or N/M	M bun no N	M分のN
c)	Ordinal Numbers		
	Number + Bam-me	番号 + 番目	
d)	Percentage		
	Unit	Name	Kanji
	1/10	Wari	割
	1/100	Bu	分
	1/1000	Rin	厘
	e.g. gowari	rokubu	nanarin = $\frac{567}{1000}$
	5割	6分	7厘

Figure 4.28: Emperor eras and Numbers

Era	From	To	Duration	Tenno
Meiji	1868	1912	45	Mutsuhito
Taisho	1912	1926	15	Yoshihito
Shouwa	1926	1989	64	Hirohito
Heisei	1989	Akihito

Table 4.6: Japanese eras

Argentina	22.42.00
Denmark	22.42.00,03
Italy	22.42.00,030
Belgium	22:42:00,030
Greece	22:42:00.030
Japan	22:42:00

Table 4.7: Time formats

a) Gengou(Nengou) 元号(年号) Calendar as Kanji

b-1) 92-4-16 ; 1992-4-16 ; 4-4-16 ; H4-4-16

b-2) 4 Nen 4 Gatsu 16 Nichi

4年4月16日

c) Heisei 4 Nen 4 Gatsu 16 Nichi

平成4年4月16日

d) Like c) with Month as Kanji

平成4年四月16日(1)

平成4年卯月16日(2)

e) Like d) with Year & Day + Month as Kanji

平成四年四月十六日 (1)

平成四年卯月十六日 (2)

Figure 4.29: Date representation

Month	Name	Name in Kanji	Traditional Name	Traditional Name in Kanji
January	Ichigatsu	一月	Mutsuki	睦月
February	Nigatsu	二月	Kisaragi	如月
March	Sangatsu	三月	Yayoi	弥生
April	Shigatsu	四月	Uzuki	卯月
May	Gogatsu	五月	Satsuki	皐月
June	Rokugatsu	六月	Minazuki	水無月
July	Shichigatsu	七月	Fumitsuki	文月
August	Hachigatsu	八月	Hazuki	葉月
September	Kugatsu	九月	Nagatsuki	長月
October	Jugatsu	十月	Kannazuki	神無月
November	Juichigatsu	十一月	Shimotsuki	霜月
December	Junigatsu	十二月	Shiwasu	師走

Table of Japanese Names for the Month

Day	Name	Name in Kanji	Abbreviation	Abbreviation in Kanji	Meaning
Monday	Getsuyobi	月曜日	Getsu	月	Moon
Tuesday	Kayobi	火曜日	Ka	火	Fire
Wednesday	Suiyobi	水曜日	Sui	水	Water
Thursday	Mokuyobi	木曜日	Moku	木	Wood
Friday	Kin'yobi	金曜日	Kin	金	Gold
Saturday	Doyobi	土曜日	Do	土	Earth
Sunday	Nichiyobi	日曜日	Nichi	日	Sun

The Names of the Day in Kanji

- f)
- | Input
Romaji | me | i | ji | Kanji |
|---------------------|----|---|----|-------|
| Display
Hiragana | め | い | じ | 明治 |

Figure 4.32: Time representation

4.5 Kana & Kanji Input

It is understandable that the Japanese can not build keyboards with thousands of keys to enter their Kana and Kanji characters (old typewriters where build in a similar way, see figure 4.34 on page 121). For this reason they use different methods to enter these characters. Usually the input is handled by a program know as FEP (Front End Processor). This program accepts the user input and handles the necessary conversions to the appropriate codes. In the following sections I will describe some of these methods and the historic development to the status quo today.

4.5.1 Front End Processor

Nowadays there are different ways to enter Kana and Kanji symbols. In this section I would like to introduce the reader to some possible ways.

Today there are two common used ways :

- Romaji \Rightarrow Hiragana \Rightarrow Kanji
- Hiragana \Rightarrow Kanji

Both ways are available in different implementation (e.g., IBM, Ricoh, NeXT, Hitachi, ...), but the basic principal behind the function is always the same. The main difference between the different methods is that some companies (or their FEP) use a separate window or line on the screen to display the Kana or Kanji before the user accepts the presented character. After the user has accepted the offered choice, the characters will be transferred to their place in the application and displayed at the last cursor position.

The other widely used method is to display the input direct on the screen (at the actual cursor position) in the application. So that the user will see his input appearing exactly at the place where he wants to enter the characters (inline conversion).

Kana to Kanji Conversion

The first way of conversion (via Romaji) is just a step before the Hiragana to Kanji conversion. For the most people this way is easier to use because they are more familiar

with the normal standard " QWERTY " computer keyboard then with the Hiragana computer keyboard (see figure 4.35 on page 122, [28]). Hiragana (and Katakana, both are called Kana characters) is a syllable alphabet, so that it is possible to enter the syllables in Romaji (our alphabet), i.e., the Latin alphabet (a table with Romaji and the corresponding Katakana and Hiragana syllable is starting at page 127). After typing the syllables in Romaji the FEP converts the input and displays the appropriate Kana character. To enter the name of the city " Tokyo " by using the first way, we have to type " toukyou " (which is the correct spelling for Tokyo in Japanese). If we type this on a " QWERTY " computer keyboard and the FEP is switched to the Romaji to Hiragana conversion mode, on the screen would appear the spelling for " Tokyo " in Hiragana (see figure 4.32, page 114, picture e).

Depending on the FEP conversion routine we could now take over the Hiragana characters by pressing a certain key (e.g., space, enter) or we could press the key which would start the conversion to Kanji characters.

Tokyo (or *toukyou*) converted to Kanji causes no problems, because there is only one Kanji which represents this spelling. After pressing the conversion key, on the screen would appear the Kanji characters for " Tokyo " (see figure 4.32, page 114, picture e).

In the other case, if there are different Kanji characters for a certain spelling than we have to choose the right one. There are different ways to do this :

- by pressing the Kanji conversion key again and again until the right Kanji will appear, or
- requesting a table with all, for this spelling, appropriated Kanji characters from the system and then choosing the right one

To make it easier for the user, the most hardware vendors have developed a special set of keys which are added to a normal keyboard. A user is now able to select the FEP mode by pressing one key or a combination of shift, control or alt(ernate) and a special key to switch between the FEP modes. To give you an example you will find in figure 4.37 pictures of a Japanese IBM notebook model 55 and the corresponding German IBM model. If you direct your attention to the space key you will recognize that the Japanese space key is much smaller then the space key on the German

keyboard. Instead of a big space key you will see three keys which are used to switch between the different modes of the FEP.

One of this keys is the Kanji character conversion key. The other keys are used to switch between Hiragana, Katakana, ASCII and other input methods (like input by JIS code).

Let me give you an example for the use of this, if you want to enter the name of the Meiji era you have to type " meiji ". On the screen would appear the spelling of " meiji " in Hiragana (see figure 4.32, page 114, picture f).

If you press now the Kanji conversion key the system would offer the most common (or often used) Kanji characters for this spelling (see figure 4.32, page 114, picture f). This could be (or is) the right Kanji character for that case, but in some cases you need an other Kanji which also represents the spelling, but has a different meaning. These Kanji characters would look like picture a) in figure 4.38 on page 125 (for the input " meiji "). Some systems are able to display a table of all appropriate Kanji characters and the user can select the right one by pointing with a mouse cursor or high-lightening the Kanji character under cursor control.

The other possible way of entering the syllables " to u kyo u " for " Tokyo " is to type them direct on the Hiragana keyboard (see figure 4.35, page 122). But today the most people prefer the " QWERTY " type computer keyboard.

Development of FEP

When the Japanese started to develop the FEP method they used a simple architecture. After the user entered the spelling of a Kanji character the FEP is looking in a Kanji database for the Kanji which fits this spelling. After the user has selected and accepted the offered Kanji characters the code of this Kanji character is passed through to the application.

The first versions of FEPs where only able to convert one Kanji character at a time. In order to get the Kanji characters for " Tokyo " the user had to type " tou " and convert this to the Kanji character for east. After that he had to enter " kyou " and convert this to the Kanji character for capital city (see picture a, figure 4.36, page 123). This, Tan (single) Kanji conversion called, way of entering characters was slow and not very sophisticated. The next step was that the FEP system was able

to understand the spelling of compound Kanji characters (like ” Tokyo ” which is a combination of the two Kanji characters for east and capital city). The Jukugo (Kanji compound) conversion enabled the user to enter ” toukyou ” and to convert this to the appropriate Kanji characters (see picture b, figure 4.36, page 123). The next improvement of the FEP was the Renbunsetsu (phrase) conversion which allows the user to enter whole phrases (or sentences) and convert them at once. This way of converting Kanji and Kana characters is quit difficult because the Japanese use not only Kanji characters in a sentence. For example, Hiragana is used for grammatical constructs and Katakana or Romaji is used for foreign words. The Renbunsetsu FEP must be able to judge which Hiragana characters have to be converted and which not. Furthermore, if there are several Kanji characters for a spelling, the system has to give the user a choice of Kanji characters (see picture c, figure 4.36, page 123) depending on the context of the sentence. This is quite more complicated and needs an intelligent conversion algorithm, much more intelligent then the conversion algorithm for compound Kanji character conversion. Today it is possible for an application to overtake control in which mode the FEP is. The FEP could be switched to ASCII mode if the application (or the programmer) wants that the user could enter only ASCII characters (e.g., for filenames). Likewise it can allow to user to choose which type of characters he wants to enter.

Structure of FEPs

The basic structure behind all FEPs is similar. Some are a part of the keyboard driver, some work as a daemon in the background of a system. But all work in a similar way (for the basic structure see figure 4.33, page 118).

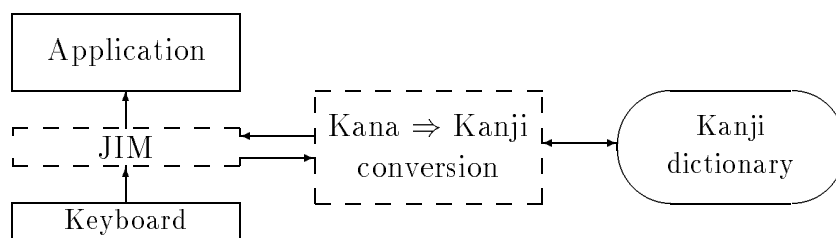


Figure 4.33: Japanese Input Manager

In order to do their job, converting a user input to Kanji character(s), the FEP needs a Kanji dictionary. In this dictionary is the pronunciation (Yomi in Japanese) for the Kanji characters listed. Depending on the mode the **J**apanese **I**ntput **M**anager (JIM) passes the user input through to the application or converts the input to Hiragana, Katagana or (via the Kana characters) to Kanji characters. Let us now have a closer look on the conversion from Romaji or Hiragana to Kanji. The normal way is that the system, after you entered the Romaji or Hiragana pronunciation, has a look into the Kanji character database to find an appropriate Kanji (or a link to the right code number of a Kanji or the Kanji characters which fit the spelling). After that the system offers you the Kanji character(s) and waits for a decision of the user.

The NeXT computer, which was japanized by Canon, uses a " Kana Kanji Conversion Server " which runs as a separate process in the background. The JIM from NeXT computer (Canon) got several different modes :

- Romaji (ASCII) input
- Hiragana input (via Romaji or Hiragana keyboard)
- Katakana input
- Conversion to Kanji characters

If the JIM is switched to the " normal " ASCII input mode it works like a ASCII Keyboard, i.e., it is passing the input through to the application.

If the mode is switched to the Hiragana mode the JIM will try to interpret all key-strokes as Hiragana. First the entered character is displayed in Romaji. If a single character represents a Hiragana syllable (like a, i, e, o ,u) it will be displayed on the spot as Hiragana. If there are several choices (like sa, se, si, so, su) the JIM will display the first Romanji and wait until it is possible to decided which Hiragana it meant (see figure 4.40, page 127). Then the appropriate Hiragana will be displayed (instead of the Romaji characters, see picture a, figure 4.39, page 126). In the Katakana mode the system will do excatly the same, only that now Katakana is selected and displayed (see picture b, figure 4.39, page 126). If the computer is connected to a Hiragana keyboard the minor difference is that now there is only one key to press, which represents a Hiragana character.

This mode of conversion has the advantage that a user could easily enter the Yomi of a Kanji. Then let this input convert to the appropriate Kanji characters by the conversion routine. However, as you maybe have recognized, you have to know the correct Yomi of the Kanji which you have in mind (like "Tokyo" is spelled "Toukyou" in Japanese). This leaves us with the problem that the Japanese know and use different systems for transliterating their Yomi to Romaji. As an example you will find a table with the Hepburn and Nippon-siki (Japanese system) starting from page 69.

Entering Katakana

The other Japanese syllable alphabet, Katakana, which is mainly used to describe foreign words or terms is entered in the same way as Hiragana. After switching to the Romaji to Katakana conversion mode it is possible to type in the syllable on the computer keyboard (see picture b, figure 4.39, page 126 or picture d in figure 4.38 on page 125).

Katakana characters are usually used to write foreign word (like names) or so called Gairaigo (loan-word from other languages). A foreign name like Schilke (spelled as Shi³ ru⁴ ke) would look like picture e, figure 4.38 on page 125.

The spelling in Katakana depends on the pronunciation of the foreign word. Following the pronunciation, the word will be represented in the "best fitting" spelling, which is possible with the Katakana syllables. Sometimes foreign words become a different spelling and pronunciation compared with the original spelling and pronunciation, e.g., the word "software" will become "sofutoueā" which looks in Katakana like picture f, figure 4.38 on page 125.

Besides these methods some systems also offer a way to enter a number code which represents the appropriate Kana or Kanji character. For the code number is the code of the character, e.g., in the system internal code, the standard JIS Ku-ten code, the JIS code or the IBM PC code (see figure 4.42 at page 129). This is only a short list, depending on the system there will be some other codes available (like Shift-JIS, ...).

³sounds like the SCHI*lke* in my name

⁴There is no L available in the Kana syllable alphabet, so that the Japanese us RU instead of L

Old Japanese Typewriter



Figure 4.34: Photos from the "Deutsches Museum Muenchen"

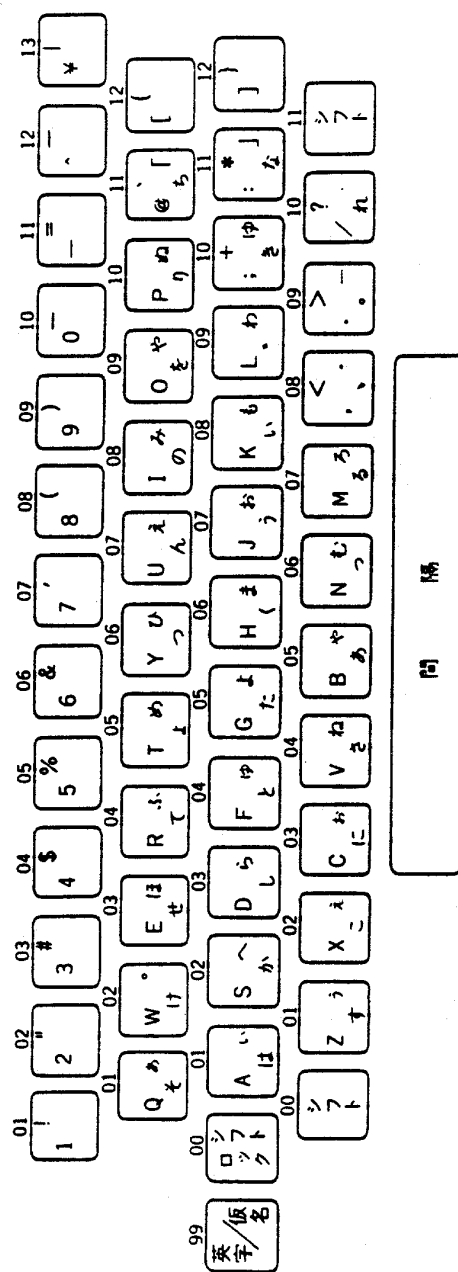


Figure 4.35: Hiragana Keyboard

Input	Display	After Conversion
tou	とう	東
kyou	きょう	京
tou + kyou	=	東京

A) Tan Kanji (single Kanji) conversion

Input	Display	After Conversion
toukyou	とうきょう	東京

B) Jukugo (Compound) conversion

Input	Display	After Conversion
toukyouha、 subarashiima chidesu。	とうきょう は、すばらし いまちです。	東京は、素晴 らしい街で す。

C) Renbunsetsu (Phrase) conversion

Figure 4.36: FEP conversion development



Figure 4.37: Japanese and German IBM PC55

- a) 1st Choice Kanji for Meiji 明示
- b) All Other Choices for Meiji 明示、明治、名辞
- c) bi, hi, ni, pi, ka, jitsu, nichu, nitsu, tachi, → 日
 び、ひ、に、ぴ、か、じつ、にち、につ、たち → 日

1 Kanji with 9 ways of Yomi

- d) Tokyo in Katakana

Input Romaji	to	u	kyo	u	Kanji
Display Katakana	ト	ウ	キョ	ウ	東京

- e) Schilke (Shiruke) in Katakana

Input Romaji	shi	ru	ke	Kanji
Display Katakana	シ	ル	ケ	知家

- f) Software (Sofutouea) in Katakana

Input Romaji	so	fu	to	u	e	a
Display Katakana	ソ	フ	ト	ウ	エ	ア

Loan word (Gairaigo 外来語)

Figure 4.38: Kanji and Katakana characters

a)

Input ASCII	JIM	Display
H	H	H
Hi	<u>Hi</u>	ひ
Hir	<u>Hir</u>	ひr
Hira	<u>Hira</u>	ひら
Hirag	<u>Hirag</u>	ひらg
Hiraga	<u>Hiraga</u>	ひらが
Hiragan	<u>Hiragan</u>	ひらがn
Hiragana	<u>Hiragana</u>	ひらがな
Conversion	<u>Hiragana</u>	平仮名(Kanji)

b)

Input ASCII	JIM	Display
K	K	K
Ka	<u>Ka</u>	カ
Kat	<u>Kat</u>	カt
Kata	<u>Kata</u>	カタ
Katak	<u>Katak</u>	カタk
Kataka	<u>Kataka</u>	カタカ
Katakan	<u>Katakan</u>	カタカn
Katakana	<u>Katakana</u>	カタカナ

Figure 4.39: JIM conversion

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
a	ア	あ
i	イ	い
u	ウ	う
e	エ	え
o	オ	お
ka	カ	か
ki	キ	き
ku	ク	く
ke	ケ	け
ko	コ	こ
sa	サ	さ
shi	シ	し
su	ス	す
se	セ	せ
so	ソ	そ
ta	タ	た
chi	チ	ち
tsu	ツ	つ
te	テ	て
to	ト	と
na	ナ	な
ni	ニ	に
nu	ヌ	ぬ
ne	ネ	ね
no	ノ	の

Figure 4.40: Kana Characters (Part 1)

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
ha	ハ	は
hi	ヒ	ひ
hu	フ	ふ
he	ヘ	へ
ho	ホ	ほ
ma	マ	ま
mi	ミ	み
mu	ム	む
me	メ	め
mo	モ	も
ya	ヤ	や
yu	ユ	ゆ
yo	ヨ	よ
ra	ラ	ら
ri	リ	り
ru	ル	る
re	レ	れ
ro	ロ	ろ
wa	ワ	わ
wo	ヲ	を
n	ン	ん

Figure 4.41: Kana Characters (Part 2)

Input Method	Input	Display	After Conversion	Meaning
Katakana to Kanji	ヘンカン (henkan)	ヘンカン	変換	Conversion
Hiragana to Kanji	へんかん (henkan)	へんかん	変換	Conversion
Romaji to Kanji	kanji	kanji	漢字	Kanji
Internal Code Input	xb5ad	xb5ad	記	Notes
JIS Ku-ten Code Input	jb5ad	jb5ad	点	Point
JIS Code Code Input	1706	1706	右	Right
PC Code Code Input	8EFA	8EFA	囚	Prisoner

Different Input Methods for Kanji Conversion

Figure 4.42: Different Input Methods

4.6 Minor Cultural Differences

The highest stage of Japanization is to support all the smaller cultural differences. On the next couple of pages you will find some of the Japanese minor differences. It is not very important to support all of this, but the support of cultural depending things could make the point which let a Japanese customer chose a foreign software product.

4.6.1 Writing Style Specialities

The Japanese can use different writing directions (see figure 4.43). A common used style of writing is the Western writing style. Additional to this writing style there are some other writing styles used. The most common way of writing in Japan is still the vertical writing direction but also Western style writing is quite often used. Vertical and right-to-left writing is not a high demand for japanized versions of foreign software. Nowadays the Japanese get used to the Western style of writing. Right-to-left writing was used in ancient times (until WW2) and today there is not really a demand for this writing style. It would be a nice feature, but it is usual not needed. The Japanese vertical style which writes from the left to the right is also very rarely used today.

Today the Japanese vertical writing style (from the right to the left) is mainly used for books and magazines (see figure 4.43). Only if you plan to sell a desktop publishing program you have to consider to enable your program for vertical writing. Besides the vertical writing the Japanese style of writing has some other differences to the Western style of writing. The page order is an other difference to Western styled books or publications (see 4.44, 4.45, 4.46). So that Japanese open their books at the, for Westerners, wrong end. They read it from the back cover to the front cover. This also causes that the page numbering is, for Westerners, in reverse order. To show you an example you will see in figure 4.47 on page 147 a Japanese poem in Western style writing (picture A) and in picture B the same poem in Japanese writing style. The Japanese writing style implicate some other Japanese specialities, like Keisen, Kinsoku Shori, Kansuji, Hankaku and Zenkaku Characters, Rubi (top and bottom rubi, e.g., also called furigana), Amikake, Kinto-waritsuke, Japanese Punctuation, small Kanas, the Japanese format for addresses and telephone numbers. In addition

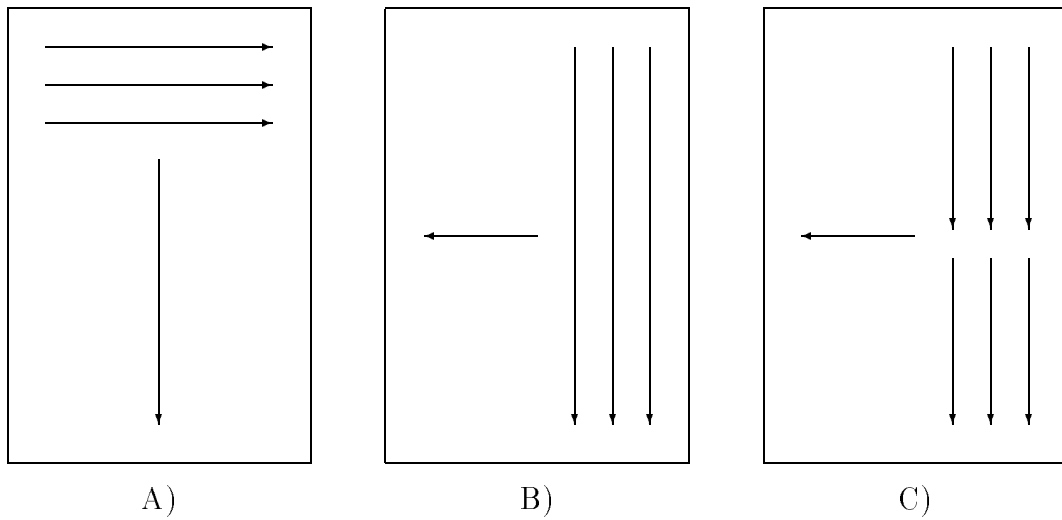


Figure 4.43: Japanese Writing Directions :

A) Western writing style (from left to the right, from top of the page to the end of the page)

B) Japanese vertical writing style (from the top of the page to their bottom, from the right side of the page to the left side)

C) Like B) with paragraphs

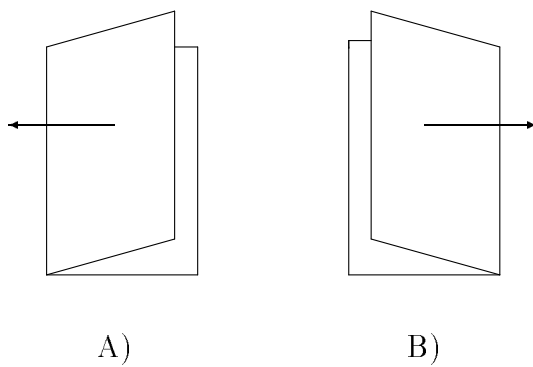


Figure 4.44: Japanese Page Order:

A) Western page order, the publication is opened from the right side

B) Japanese page order, the publication is opened from the left side

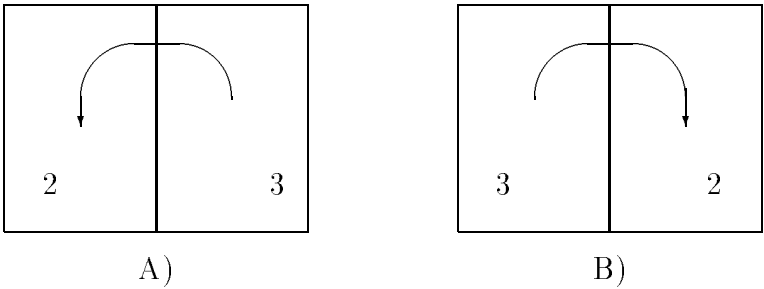


Figure 4.45: Japanese Page Order:

- A) Western page order, the publication is leafed through by turning over the right page
- B) Japanese page order, the publication is leafed through by turning over the left page

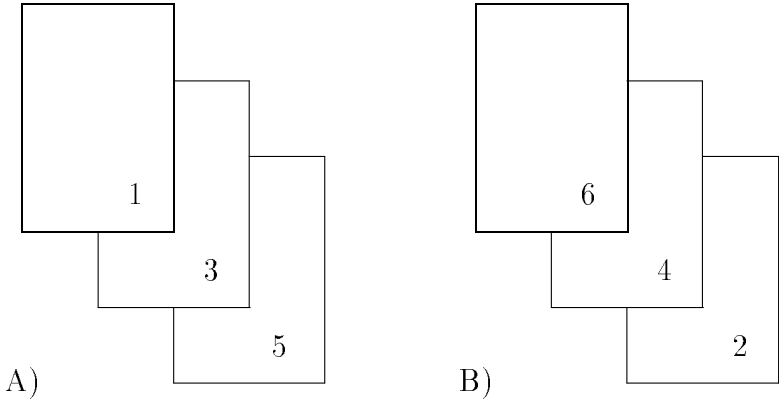


Figure 4.46: Japanese Page Order:

- A) Western page order, the pages are sorted 1,2,3,4,5,6,...
- B) Japanese page order, the pages are sorted (for us in reverse order) ...,6,5,4,3,2,1

there is a difference in the use (or understanding) of symbols for yes & no.

Keisen

Keisen are used for creating tables, lists or reports. Keisen are, literally translated, lines, boxes or borders. Vertical lines are called Tatesen and horizontal lines are called Yokosen. Keisen are not a requirement for all programs. When your program produces output of data in form of tables, lists or reports it will be considered as necessary to provide Keisen.

Kinsoku Shori

Kinsoku Shori is similar to the word wrap in Western word processors. It adjusts text that specific characters do not appear at the start (comma, Japanese period or symbols, see picture C in figure 4.48, page 148) or at the end of a line (open parentheses and quotation marks, see picture C in figure 4.49, page 149). This is called line head Kinsoku processing (see pictures A & B in figure 4.48) and line end Kinsoku processing (see pictures A & B in figure 4.49). Usually these characters are moved to the previous or next line.

Kansuji

Kansuji is the traditional Japanese way of writing numbers. Besides the (adapted) way of writing numbers in Western style (like 1, 12, 123, 1234, ...) the Japanese use their traditional way of writing numbers. Especially when the numbers or numbers and text are written vertical Kansuji is used. In addition to "normal" numbers the Japanese use special Kansuji to indicate the numbers 10, 100, 1,000, 10,000, 100,000,000 and 1,000,000,000. On page 103 in figure 4.27 you will see the Kansuji and an example of their use. The only difference between writing numbers in Kansuji and Kanji is that the numbers 0, 1, 2, 3 have a different Kanji.

Hankaku and Zenkaku Characters

The terms Hankaku and Zenkaku refer to the width of characters. For example Katakana is not always displayed in Zenkaku, which is the doubled width of normal

ASCII characters. If Katakana is displayed in Hankaku it needs the same space on the display as a standard ASCII character needs. For example on an AX-PC the size of a Hankaku character is 8×19 dots and the Zenkaku characters is 16×19 (in a 24×24 matrix) dots in size. Only ASCII and Katakana are displayed in Hankaku. Depending on the code set it is represented by one or two bytes.

Rubi

Rubi is another Japanese specialty. Rubi's (sometimes called furigana) are used to write the pronunciation (Yomi) of a rarely used Kanji on top (top Rubi, see picture A in figure 4.50 on page 150) or underneath (bottom rubi, see picture B in figure 4.50) this Kanji. They have usually the half size of Zenkaku characters .

Amikake

Amikake is used like boldface, capitalizing or underlining text. It is somewhat like gray screening of text or characters (see picture C in figure 4.50 on page 150).

Small Kanas

Small Kana characters are used to distinguish between different Yomi's of Kana characters. The character looks like a normal Kana, it is just smaller. Added to a normal Kana character it changes the Yomi of this character, e.g., Do becomes Dyo by adding a small Kana character (see picture D in figure 4.50 on page 150).

The other special Kana characters are called Dakuon, Yo'on, Sokuon and Handakuon. These are Kana characters which express different pronunciations of the standard Kana character. Dakuon are distinguished from normal Kanas by the Japanese version of the ". These characters are called "voiced sounds" (see figure 4.51 on page 151). Yo'on is a, so called, "contracted sound" and is also expressed by small Kana characters (see picture A in figure 4.52 on page 152). Sokuon is actually only one character (tsu) which is a small character and stands for a "double consonant (or assimilated) sound" (see picture B in figure 4.52). The last group of special Kana characters is the Handakuon group. This group has the Japanese ° as sign. All Handakuon start with " P " (see picture C in figure 4.52) and therefor they are called P-sounds.

Kinto-waritsuke

Kinto-waritsuke is a special way of formatting Japanese characters or sentences between two points. It is used to stretch text between, e.g., the beginning of a line and the end of a line (see pictures A & B in figure 4.53 on page 153).

Japanese Punctuation

The Japanese use several types of punctuation which are not common or unknown in the Western world. Some of them just look different from their Western counterpart. There are several different types (see picture C in figure 4.53, page 153) :

- the Japanese period
- the Japanese comma
- the Japanese separator
- the Japanese repeat symbol
- other punctuations like Japanese parentheses, brackets, ...

Besides the separator and the repeat symbol they are used like Western cultures would use this punctuation. The main difference is the special shape.

The separator is used to make it easier to understand compounds of Kanji's so that it is easier to understand the meaning of a certain Kanji group. The repeat symbol indicates that the previous Kanji has to be repeated.

Address Format

The Japanese address format is totally different from the address formats used in the Western societies. The way of writing an address in America would look like this :

Addressee's Name

Number Street

City State (or State abbreviation) Zip Code

An address on a German letter would look like this :

Addressee's Name
Street Number
W- (or O- for the former East German area) Zip Code City

The Japanese use a reverse order compared to this two Western styles :

Postal Symbol Zip Code
City (Town or Village)
District Number
Addressee's Name

If your program prints anything with an address on it (invoices, letters, ...) you have to enable your program to cope with this style of Japanese Address.

Telephone Numbers

Phone numbers are, like the date, written in various formats. In addition to the western way of writing phone number (grouped in 3 digit or 4 digit groups) like :

- (03) 3479-2893
- 03-3479-2893

there is a way of writing the number in vertical writing as you will see in figure 4.54 on page 154. As well as the use of Kansuji it is also common to write a phone number vertical with western numbers.

Yes / No

An interesting difference between Western culture and Japanese culture is the way of expressing Yes and No. For a person with a Western background an **X** usually represents Yes and a circle (○) represents NO. In Japan these symbols will be

Product	CGA	Hercules	VGA
Chicago Software	○	○	X
Dallas Software	○	X	X

Table 4.8: Demo Table for Japanese Yes / No

interpreted in the opposite way. In a Japanese Table ○ represent Yes and the **X** represents NO. As an example let us have a look at table 4.8. An European or American reader would interpret this table in the way that Chicago Software supports only VGA and Dallas Software supports VGA and Hercules. A Japanese reader, instead, would assume that Chicago Software supports CGA and Hercules and Dallas Software only CGA.

In addition the way of answering is also different. If the system asks you " Don't you want to delete this file ? Yes / No " a Japanese user would answer " **YES**, I don't want to delete it " ([11])

4.6.2 Other Differences

Further to the differences mentioned above there are plenty of other differences in the Japanese (language & cultural) computer environment. I can not list all of them but I will give you some other differences which could apply in a software product.

For example : if you want to adapt a CAD program for the Japanese market or if your program just has to make some paper output you should know which units and which paper size is used in Japan. Sometimes it is useful to know which differences between foreign and Japanese hardware exists. Or think about the problem how to sort Japanese data. Even if you do not need to sort Japanese data you need at least a data-type to store the data.

Units

The Japanese use, unlike the English or American, mainly the metric system ([4]). To express a length they use meter as a base unit (also depending forms like millimeter,

centimeter, kilometer, etc.) and the cubic or square form of the base unit for area and volume (see tables a, b & c in figure 4.55 on page 155).

Another form to express a volume is the base unit liter. For the measurement of weight the base unit gramme is used. The base unit for time is the second. To represent a time hour, minute and second are used (see tables a & b in figure 4.56 on page 156). For temperature the Celsius scale is used ([4]).

Maybe you have recognized the "mainly" above. This implies that the Japanese use not only the units from the metric system. In addition to metric system the Japanese use some of their traditional units for measurement purposes. You will find them in the figures 4.55 and 4.56. For example real estate agents often use the traditional measures for area to describe the size of an apartment or room (e.g., Tsubo).

Paper size

Similar to the unit system, the Japanese use the same paper sizes as, e.g., Australia, Germany, etc. In table 4.9 you will find the sizes of the standard which is used in Japan. This standard corresponds, e.g., to the German DIN 66008. Additionally you will find some of the American paper sizes (letter and Legal).

The most common used paper sizes in Japan are A4, B4, B5 and A3. For normal business communication A4 is mainly used. Some governmental forms are printed on B sizes.

4.6.3 Hardware

Regarding the fact that a Japanese computer user has several special requirements it is naturally that there are some differences in the hardware. The major differences are caused by the fact that the Japanese use Kanji characters. This causes certain requirements, like :

- Kanji ROMs for the Kanji fonts & character set (nowadays sometimes realized as softROMs)
- high resolution for the output on screen and/or printer usually a Kanji character is coded in a 24×24 matrix (this was the main reason for the development of the 24 (!) wire printer and laser printers).

Size	JIS (DIN) A †	JIS (DIN) B †	JIS (DIN) C †
0	841×1189	1000×1414	917×1297
1	594×1189	707×1414	648×917
2	420×594	500×707	548×648
3	297×420	353×500	324×458
4	210×297	250×353	229×324
5	148×210	176×250	162×229
US Letter	215×279 †		
US Legal	215×355 †		

Table 4.9: Standard Paper Sizes used in Japan

† = in mm

Source : [1], 7-1

- a keyboard which supports Kanji character input with special conversion keys (not really necessary, but more convenient for the user)
- support of the JIS Kanji character set standard, e.g., the most printers work with this standard (as control language the Japanese version of the Epson ESC/P printer control language is widely used)
- in the Japanese PC world exists a different disk format in addition to the IBM PC standard formats. This format is 1.2 MB on a 3.5 inch floppy disk (Disk-drives which supports this format must be able to change the speed of the disk-spin).
- the size of a computer system is more important in Japan than it is in Europe or the USA. As mentioned before, the office (and private) space in Japan is limited. This creates a demand for small, but powerful, computer systems (for a picture of a compact office computer system see figure 4.57 on page 157).

These are only the obvious differences, but it gives you a fair impression about it. In the workstation world the differences are not so big and mostly solved by the use of software (like softROMs, softfonts, ...)

Let me use as an example the, so called standard in the western computer environment, the IBM PC (or compatible). Compared with the NEC PC98XX series, the market leader in Japan, you will spot some significant differences ([22]). Both machines run under the operating system MS-DOS (or PC-DOS). A "clean"⁵ DOS program which uses only DOS calls will probably run and also "clean" Windows programs are usually compatible.

If a program relies on PC BIOS⁶ calls the it will not run on a NEC PC. In addition the DOS of the IBM PC is a standard ASCII (IBMSCII) OS which supports only some European character extensions. The NEC PC instead runs a full Kanji version (using Shift JIS) of MS-DOS.

The most differences are based in the different hardware design of the NEC PC. These differences are :

- Keyboard layout, in order to make it easier for the Japanese user NEC added some special keys for the use with the FEP, some special application function keys and renamed some other keys. Another difference is that the NEC PC uses a FEP to handle the Kanji input (compared to the US IBM PC). Starting from page 157 you will find some examples for different Japanese keyboard layouts.
- Floppy disks are compatible in some way. The NEC PC can read the IBM PC disks in the formats 360KB (5.25"), 1.2MB (5.25") and 720KB (3.5"). In addition the NEC is able to write IBM PC disks in the 1.2MB and 720KB formats. It is not possible for the NEC to access the 1.44MB (3.5") formatted disks or to write on a 360KB floppy disk. The format which is only supported by Japanese computers is the 1.2MB (3.5") format. The IBM PC is not able to read or write this Japanese format because the system must be able to switch the speed of the disk spin.
- The Video memory has a totally different design. Instead of one byte for a character the NEC always uses two bytes. Not only the way of storing a character is different also the display attributes have a different organization then the IBM PC display attributes.

⁵using only the specified DOS calls or Windows APIs

⁶Basic Input Output System

- When NEC launched the NEC PC it was a requirement to work with Kanji characters. To do this they needed a higher resolution than the IBM PC was offering at this time. The NEC PC resolution in normal mode is 640×400 pixel and the high resolution mode it is 1120×750 pixel. This specification has not changed in the last ten (!) years.
- The NEC PC Kanji fonts are stored in a ROM chip. Nowadays the IBM PC stores them in the PC memory.
- The BIOS calls are the main hurdle for foreign software or programmers. In table 4.10 (on page 141) you will see the main differences between the IBM PC BIOS and the NEC PC BIOS. In addition the BIOS RAM area (which keeps

IBM PC BIOS	BIOS Call	NEC PC BIOS
09_{Hex} & 16_{Hex}	Keyboard	18_{Hex}
10_{Hex}	Video	18_{Hex}
13_{Hex}	Disk	$1B_{Hex}$
14_{Hex}	Serial Communications	19_{Hex}
15_{Hex}	System Services	various
17_{Hex}	Printer	$1A_{Hex}$

Table 4.10: IBM vs. NEC BIOS calls

([22])

the system status, keyboard buffer, information about the graphics mode) has a different layout.

These are the main differences. On the Japanese market there are many other players which have their own hardware and software design (like Fujitsu, Toshiba, ...).

In the workstation environment the differences are not so big. The most differences could be handled by software. The main difference in the hardware is the Japanese keyboard.

4.6.4 Sorting

It is quite easy to sort data containing alphabetic or numeric data. Even if you have special characters like German umlauts (e.g., the "a is treated as ae, it is possible to use a normal alphanumeric sort routine). For the most European or alphabet oriented countries there is a definition for the collating sequence, but now ask yourself : How would you sort pictures ? In an easy way you could look at a ideographic Kanji character as a picture. Now you should be able to imagine how difficult it is to sort Japanese data. There are several approaches to sort Japanese data, like :

Through the fact that you could compare Hiragana and Katakana with an alphabet (a syllable alphabet) it would be possible to sort data using the Yomi of the, e.g., name, instead of the ideographic Kanji characters which represent the name. To do this you have to store for each data-field (in a separate field) the pronunciation. The storing of the Yomi is necessary because a Kanji character has not only one, sometimes several, Yomi's. The pronunciation also depends on the way of reading the Kanji character. The most common ways are On-yomi (Chinese) and Kun-yomi (Japanese) reading (or pronunciation) This makes it much easier to sort Japanese data, but the problems just have begun. The main problem is that there is no Japanese standard ordering (or collating sequence) defined. This leaves us, again, with the problem how to sort Japanese data.

To sort data represented in Hiragana characters you have to use a collating sequence like the 50-On-Sequence (starting a-i-u-e-o (instead of a-i-e-o-u in the western world), called Gojuuonjun) or the I-RO-HA collating sequence (called after an old Japanese poem. It starts I-RO-HA and all Hiragana characters appear just once). A table (see figure 4.60) with these collating sequences starts from page 160. By using one of these sort sequences you also have to distinguish between sorting after the actual Yomi, the On-yomi or Kun-yomi. Furthermore this way of sorting Japanese data, there are several other ways of doing this. The disadvantage of the method described above is that you always have to store the Yomi for the sorting process. If you want to avoid this you have to use one of the other methods of sorting Japanese data.

Other ways of sorting Japanese data are, e.g., sorting after the numbers of strokes of a Kanji character (this collating sequence is called Sokaku), after the type of the strokes (called Bushu) or after the JIS code table (which is actually the easiest way of sorting Japanese data).

Sorting after the JIS code table brings the data in the following order ([26], [27]) :

- JIS X0208 level 1 is sorted after the representative On-Kun Yomi of the Kanji character by using the 50-On sequence. Kanji characters with the same Yomi are sorted in the order On-yomi, Kun-yomi, order of Radicals (see below), number of strokes.
- JIS X0208 level 2 is collated in the 214 classes of Radicals (see below). In these classes the Kanji characters are sorted after the number of strokes. If some characters have the same number of strokes they are sorted following the 50-On sequence.
- JIS X0212 (sometimes called JIS level 3) is sorted like JIS X0208 level 2 in the 214 groups of Radicals and within a Radical group after the number of strokes.

In the description of the collating sequence which is used for the JIS character set we find a new term called Radical. A Radical (part-head group) is an ideograph, a base which is used in combination with other Radicals to form Kanji characters. Usually a Kanji character is formed by up to four Radicals. On page 162, picture a in figure 4.62 you will find some Kanji characters which are formed with one (No. 1), two (No. 2), three (No. 3) and four Radicals (No. 4). There are several ways of combining Radicals to a Kanji character, e.g., \square , $\square\square$, $\square\square\square$, $\begin{smallmatrix} \square \\ \square \end{smallmatrix}$, $\begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix}$, ... (with \square representing a Radical).

So you see that it is not easy to sort Japanese data, but there is a way to do this. If you do not want to implement your own sort algorithm you could by a package which does this for you.

4.6.5 Japanese Data-types

In order to enable a programming language (or application) to work with Japanese characters we have to enable the system to work with DBCS characters. There are two ways of enabling a programming language or application to cope with DBCS characters.

1. Using the old data-types and enabling the system to work with DBCS characters. The Japanese version of Oracle goes this way. If you define a data-field of

20 character it is a normal SBCS data-field which could store up to 20 SBCS character. By enabling the system to work with DBCS character it is now able to store up to 10 DBCS characters or a mixed string containing any combination of SBCS and DBCS characters. In the case of a mixed string the byte length of the string can not exceed the limit of 20 byte.

2. By defining a new data-type for the handling of DBCS character (or mixed) strings. This is done, e.g., for COBOL (DISPLAY-2), FORTRAN (NCHARACTER) and C (wchar_t) ([9]). The problem with the extended data-type is that the compiler has to be rewritten because all functions handling character data have to work character oriented instead of byte oriented. The new data-type affects all types of commands, like input/output, assignment, string handling, sort/merge and comparative operations.

There are two ways of implementing a DBCS data-type. For the example I will use the widely used programming language C. The ISO 9899:1990 (ISO C) standard ([5]) defines the C data-type wchar_t. This data-type is a wide character data-type and stores the character data like 8C8E_{Hex}. In addition this standard defines routines for the conversion between multi-byte and wide characters. The second approach is to store data as a multi-byte character which looks like 8C_{Hex}8E_{Hex}. In this approach n single bytes are used to store the information. As a wide character the data is stored as a group of n-bytes (8C8E_{Hex} vs. 8C_{Hex}8E_{Hex}, [6])

The advantage of a programming environment which can handle DBCS characters is that the programmer is able to write internationalized programs. The program is able to run in different national computer environments. The emerging problem is that the programming environment has to be rewritten that it is able to handle (probably different) DBCS character sets. In addition the system must offer different conversion routines between SBCS and different DBCS (also between multi-byte characters and wide characters)

4.6.6 Japanization Pitfalls

In this paragraph I will talk about some pitfalls which could cause misunderstanding if you are not careful enough when you japanize your product.

One of the common anecdotes about a japanized product is the story about the beep in the Japanese version of Lotus 123. As mentioned before office space is rare in Japan. Many people work in open-plan offices. When Lotus launched their first version of Lotus 123J they discovered that they had to remove the error-beep. First of all it is very shameful for a Japanese when the computer tells everybody ” BEEP, you made a mistake ” and in open-plan offices the steady beeping would cause a major disturbance. As mentioned in the date section the Japanese use a special date format which contains the era of the emperor. In one of the earlier Lotus 123 versions it was possible to add the common Gengou date or the reign of the emperor and it was possible to change the name of the emperor. This was a big mistake because it looked for the Japanese that one was planning the death of the emperor ([12]).

Two other things, from the thousands of pitfalls, I will mention here are translation errors and manual design. When the first translation of the UNIX operating system took place there were some wrong translations made when they adapted UNIX to the Japanese language environment. The development of UNIX mainly took place in the US academic environment, so many UNIX-related terms have funny names (for US people) like demons, zombie process and killing a (child) process. In the first translation the message ” a child process was killed ” was translated to the horrifying Japanese sentence ” we just murder your first-born child ”.

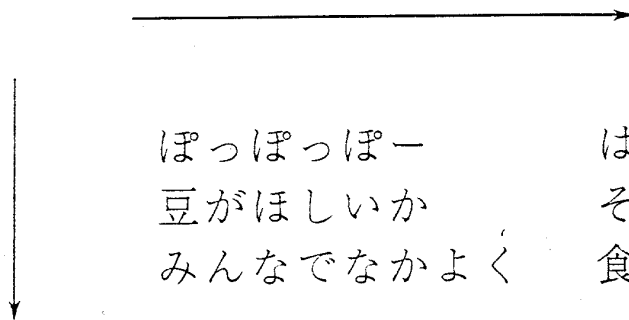
The Japanese use a lot of adapted English technical denotation but sometimes they prefer more (for them) visible Japanese terms, e.g., a Kanji which expresses an idea better or more understandable for Japanese, then the English loan-word. If you let translate the manual you should always use a Japanese native speaker for the translation. After the first translation is done you should let a second technical translator do a back translation, to check the translation of the first translator ([20]).

This is important because now you can control if the first translator gets the point which you want to express in the manual. Another fact is that the Japanese manuals have a different style than US or European manuals. In Japan the manual describes a scenario and tries to explain the user (with examples) how to work with the product. Besides that small cartoons are very common, like a floppy disk which tries to avoid the contact with a magnet (see on the cover of the most $5\frac{1}{4}$ floppy disks).

4.6.7 Cultural Differences

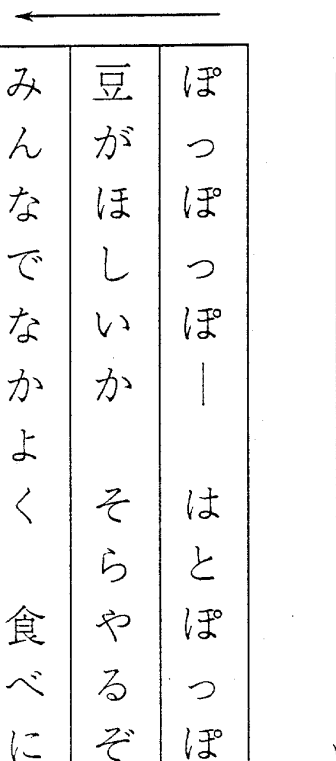
You have read now a lot about Japanese cultural differences. Not all of them have to apply for an adapted program. Nevertheless always some of them apply. If you do the japanization for your program you have to think about which of these differences you have to adapt and which not. You could start by adapting the, for your users, more important features and later on you adapt more and more of the cultural differences.

If you think you do not have adapt even some of the minor differences you will recognize that your software will not sell well (except you have a market-niche). It is like that you are offering a word-processor in Germany which is not able to handle the German umlauts. It will not be a good seller.



ぽっぽっぽー	はとぽっぽー
豆がほしいか	そらやるぞ
みんなでなかよく	食べに來い

A) Japanese poem in Western horizontal writing style



み	豆	ぽ
ん	が	っ
な	ほ	ぽ
で	し	っ
な	い	ぽ
か	か	ー
よ	そ	は
く	ら	と
食	や	ぽ
べ	る	っ
に	ぞ	ぽ
來		ー
い		

B) Japanese poem in Japanese vertical writing style

Figure 4.47:

.... AX-1とAX-2があります
 。 AX-2はスーパーインポーズが
 できますが、AX-1では.....

A) Text before Kinsoku Line Head Processing

.... AX-1とAX-2があります。
 AX-2はスーパーインポーズが
 できますが、AX-1では.....

B) Text after Kinsoku Line Head Processing

Symbol	Meaning
。	Japanese period
、	Japanese comma
,	Comma
.	Period
.	Raised dot
!	Exclamation mark
?	Question mark
:	Colon
;	Semicolon
-	Hyphen
々	Character repeat symbol
)	Close parentheses
>	
}	
」	
'	Quotation marks
”	
イ、エ、...	Lower case katakana used for loan words

C) Prohibited Symbols at line head

Figure 4.48:

．．． AXマシンには AX-1とAX-2 (スーパーインポーズ可能)の2つのタイプがあります。

A) Text before Kinsoku Line end Processing

．．． AXマシンには AX-1とAX-2 (スーパーインポーズ可能)の2つのタイプがあります。

B) Text after Kinsoku Line end Processing

Symbol	Meaning
(Open parentheses
< ≪	
[{	
{	
「 『	Quotation marks
,	
”	

C) Prohibited Symbols at line end

Figure 4.49:

あめ
雨

水
みず

A) Top Rubi

B) Bottom Rubi

これは、あみかけです。

C) Amikake

Romaji	Kana	Romaji	Kana
Do	ど	Dyo	ぢょ
Re	れ	Rye	りえ
Mi	み	Myi	みい
Do	ド	Dyo	ヂョ
Re	レ	Rye	リエ
Mi	ミ	Myi	ミイ

D) Small Kanas

Figure 4.50:

a) Dakuon だくおん 濁音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
ga	ガ	が
gi	ギ	ぎ
gu	グ	ぐ
ge	ゲ	げ
go	ゴ	ご
za	ザ	ざ
zi	ジ	じ
zu	ズ	ず
ze	ゼ	ぜ
zo	ゾ	ぞ
da	ダ	だ
di	ヂ	ぢ
du	ヅ	づ
de	デ	で
do	ド	ど
ba	バ	ば
bi	ビ	び
bu	ブ	ぶ
be	ベ	べ
bo	ボ	ぼ

Figure 4.51:

a) Yo'on ようおん 拗音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
a	ア	あ
i	イ	い
u	ウ	う
e	エ	え
o	オ	お
ya	ヤ	や
yu	ユ	ゆ
yo	ヨ	よ

b) Sokuon そくおん 促音

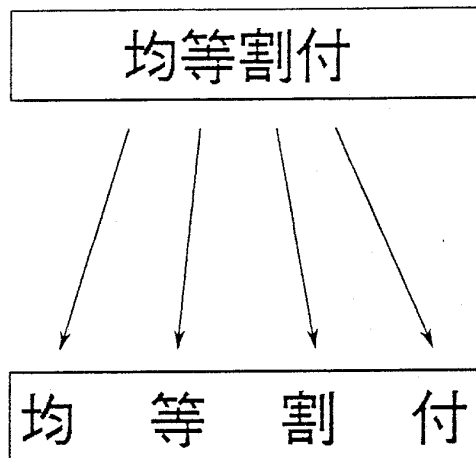
Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
tsu	ッ	っ

c) Handakuon はんだくおん 半濁音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
pa	パ	ぱ
pi	ピ	ぴ
pu	プ	ぷ
pe	ペ	ぺ
po	ポ	ぽ

Figure 4.52:

A) Text before Kinto-Waritsuke Processing



B) Text after Kinto-Waritsuke Processing

Name	Symbol
Period	。
Comma	、
Separator	・
Repeat Symbol	々
Parentheses	() []
Corner Brackets	「 」 『 』
Pointed Brackets	< > << >>
Ellipsis	...
Question Mark	?
Exclamation Point	!

C) Punctuation Symbols

Figure 4.53:

a) Telephone Number in Japanese Writing Style

Tel (03) 3479-2893

T	電
e	話
l	
(03)	(03)
3479	3479
28	28

Fax (03) 3479-5579

F	フ
a	ァ
x	ックス
(03)	(03)
3479	3479
5579	5579

Figure 4.54:

a) Length

Name	Unit	Sign
Meter	1m	m or 米 or メートル
Ri	3.9273km	里
Shaku	30.30cm	尺
Sun	3.03cm	寸

b) Area

Name	Unit	Sign
Square Meter	1m ²	m ² or 平米 or 平方メートル
Square Kilometer	1000m ²	km ² or 平方キロ米 or 平方キロメートル
Jou	1.62m ²	畳
Tan	991.7m ²	反
Tsubo	3.306m ²	坪

c) Volume

Name	Unit	Sign
Rippou Meter	m ³	m ³ or 立米 or 立方メートル
Liter	l	l or リットル
Milliliter	ml	ml or ミリリットル
cc	cc	cc
Gou	180.39cm ³	合
Shou	1.8039l	升
To	18.039l	斗

Figure 4.55:

a) Weight

Name	Unit	Sign
Gramme	g	g or グラム
Kilogramme	kg	kg or キログラム
Ton	t	t or ton or トン
Kan	3.75kg	貫
Momme	3.75g	匁

b) Time

Name	Unit	Sign
Byou	Second	秒
Hun	Minute	分
Ji	Hour	時

Figure 4.56:



Figure 4.57: .



Figure 4.58: Japanese Keyboards, Page 2



Figure 4.59: Japanese Keyboards, Page 3

Romaji ローマ字	Hiragana ひらがな 50-On- Sequence	Romaji ローマ字	Hiragana ひらがな Old-Sequence
a	あ	i	い
i	い	ro	ろ
u	う	ha	は
e	え	ni	に
o	お	ho	ほ
ka	か	he	へ
ki	き	to	と
ku	く	chi	ち
ke	け	ri	り
ko	こ	nu	ぬ
sa	さ	ru	る
shi	し	wo	を
su	す	wa	わ
se	せ	ka	か
so	そ	yo	よ
ta	た	ta	た
chi	ち	re	れ
tsu	つ	so	そ
te	て	tsu	つ
to	と	ne	ね
na	な	na	な
ni	に	ra	ら
nu	ぬ	mu	む
ne	ね	u	う
no	の	wi	ゐ

Figure 4.60:

Romaji ローマ字	Hiragana ひらがな 50-On-Sequence	Romaji ローマ字	Hiragana ひらがな Old-Sequence
ha	は	no	の
hi	ひ	o	お
hu	ふ	ku	く
he	へ	ya	や
ho	ほ	ma	ま
ma	ま	ke	け
mi	み	hu	ふ
mu	む	ko	こ
me	め	e	え
mo	も	te	て
ya	や	a	あ
yu	ゆ	sa	さ
yo	よ	ki	き
ra	ら	yu	ゆ
ri	り	me	め
ru	る	mi	み
re	れ	shi	し
ro	ろ	we	ゑ
wa	わ	hi	ひ
wo	を	mo	も
n	ん	se	せ
		su	す

Figure 4.61: 50-on Sequence and I-RO-HA

a)Radicals

- 1) 木 (Ki、き、Tree)
- 2) 林 (Hayashi、はやし、Wood)
- 3) 森 (Mori、もり、Forest)
- 4) 街 (Machi、まち、Town)

Figure 4.62: Radicals