higher rates. To explore this possibility, I analyzed data on the 15 investigators who submitted five or more abstracts each and who used both typing methods. In this subgroup, 19 of 55 regularly typed abstracts were accepted (34.5 percent), whereas 31 of 53 of the "typeset" abstracts were accepted (58.5 percent) (P = 0.015).

These results demonstrate that the new "typeset" appearance of data increases the chance of acceptance. It may mean that "typeset" printing may cause the data to look more impressive. Alternatively, it may mean that the new printing makes it easier for reviewers to read the data and to appreciate its meaning.

Most important, it means that this technological innovation reduces the chance of success of those not currently using it.

GIDEON KOREN, M.D.
Hospital for Sick Children
Toronto, ON M5G 1X8, Canada

Software

Tib: a Reference Setting Package, Update

J. C. Alexander
University of Maryland

There have been a number of minor bug fixes and some refining of features of the TEX bibliography setter Tib (see TUGboat vol. 7, no. 3, for an article about Tib). Its version number has been incremented. Those people who asked to be put on my mailing list have been sent all the changes. However, I know from mail that there are a number of other users, presumably people who picked it up via anonymous ftp. Those people might want to check the file `CHANGES` and/or `READ.ME` via anonymous ftp from `eneevax:pub/tib`. Incidentally, I appreciate the kind comments and suggestions people have made. It seems Tib is proving to be a useful adjunct to TEX.

Portuguese Hyphenation Table for TEX

Pedro J. de Rezende
Northeastern University

I have compiled a Portuguese hyphenation table for TEX. It turns out to be a rather short table (compared to the one for English) because Portuguese has very concise rules for hyphenation. I'd like to make this table public and freely distributed. Even included in the distribution tapes. I have extensively tested it (with `patgen`) and haven't found any erroneous hyphenation. It does miss some hyphens but they are very, very few. It certainly does not hyphenate a word beyond an accent or a cedilla, but that's the way TEX handles hyphenation of words with intervening macros (see Appendix H of *The TEXbook*).

Editor's note: Arrangements are being made to include the Portuguese hyphenation table in the standard distribution. Hyphenation tables for languages other than English are frequently requested on TEXhax; anyone who knows of the existence of such tables is asked to send the relevant information to Barbara Beeton, so that a list can be compiled for the next issue of TUGboat.

A (Hopefully) Final Extension of Multilingual TEX

Michael J. Ferguson
INRS-Télécommunications
Montréal, Canada

This note reports the, hopefully, final extension to TEX that allows for multilingual hyphenation reported in July 1985 (Vol. 6, No. 2, pp. 57–58) and March 1986 (Vol. 7, No. 1, page 16) of TUGboat. The key feature of the extension is that it accommodates **standard TEX fonts**, including words with accented letters. For details of the features the reader should refer to the July 1985 TUGboat. This note reports some recent extensions to accommodate certain typographical and input conventions in non-English text. These extensions are as follows:

- TEX will now hyphenate words that have an explicit `\discretionary`. Each part of the word including the discretionary is treated as a separate word for hyphenation purposes. This allows for the hyphenation of words such as "Wechselstromwecker" where the "ck" is represented by `\discretionary{k-}{k}{ck}`. The hyphens then given by `\showhyphens` are "Wech-sel-stromwek-ker".

  The discretionary hyphen approach also allows for the suppression of an unwanted ligature. This can be done by inserting a discretionary hyphen in the appropriate place. Thus the unwanted ligature in "auffrischen" is defeated with the insertion of `\-` after the first "f" to give "auffrischen". Note that the solution to exercise 5.1 in early editions of the TEXbook is incorrect as it will not survive a second pass of a paragraph. The second hyphen after the "i" remains with the extension.

  The extension is invoked by making the integer parameter `\dischyph` non-zero. Thus `\dischyph=1` will allow, on a paragraph-by-paragraph basis, hyphenation of words that have an embedded `\discretionary`. Note that by using an empty discretionary, a break is allowed without inserting a hyphen character.

- Two new integer parameters, `\starthyph` and `\stophyph`, have been defined. These allow the number of characters at the beginning and end of words that suppress hyphenation to be modified. The defaults are 2 and 3 respectively as in standard TEX. The minimum length of a word to be hyphenated is the sum of these two values. If necessary, a third independent integer parameter that specifies the minimum length of a hyphenated word could be added.

- In order to handle special keyboards with extended characters encoded outside the standard ASCII set, all characters with codes outside this set have been declared **permanently** active. This means that both the single character and the single character command may be separately defined. This would allow special discretionary sequences for various languages to be input easily.

  This extension has been in use on the VT-200 series of terminals by Digital and will work equally well for IBM-PCs. Since this modification takes effect at TEX's mouth, the extended characters never make it "inside". This means that they should not be used in definitions. For example if ü is one of these extended characters, then the definition `\Hühn{`*text*`}` actually defines `\H` and not the entire `\Hühn`. Interestingly enough, TEX will not complain. It just sticks the rest into the parameter argument.

Hopefully these extensions will be suitable for most other languages — either independently or together — assuming that the appropriate patterns and exceptions exist.

Report on JTEX: A Japanese TEX

Yasuki Saito
NTT Electrical Communications Laboratories
Japan

This is a short report of the current status of Japanese TEX, called JTEX. I do not try to give a detailed description of every nook and cranny. Instead, I will concentrate on giving the overview of what I have done to make TEX typeset Japanese text as well as English.

Example

First of all, look at the example input file and the corresponding output generated by JTEX in Appendix 1. (The input file listing was generated using the JLATEX verbatim mode.) It is an excerpt from a famous textbook on analysis written by Teiji Takagi. JTEX is an upward compatible extension of TEX and everything in TEX is at your disposal. So you will find familiar control sequences in the input file. The only difference is of course that there are lots of Japanese characters in it! Actually, from the user's point of view, the fact that he can enter Japanese characters into the input file is the main difference although he must learn a few new control sequences to select fonts and to control spacing.

Two major problems

Many people think that it is difficult to make Japanese TEX, but it is not so. There were two problems to be solved in making JTEX. One was to make TEX's input mouth a little bit wider so that it can swallow Japanese characters. The second, and more serious problem, was to prepare the fonts for more than 6000 Japanese characters.

Knuth suggests a way to extend TEX to oriental languages in "TEX: The Program", page 57. His suggestion is to extend the data structure for a character so that TEX can handle more fonts each having more characters in it. I chose not to extend the data structure nor the font file format for various reasons. A GF file with information for 6000 Japanese characters in it is just too large to maintain. Another reason is that if you stick to the original data structure you can make modifications minimum. And with the ordinary font file format you can use various utility programs without modification.

Thus I divided Japanese characters into 33 subfonts each having at most 256 characters. TEX can handle maximum of 256 fonts at a time, and reserving 33 fonts for a single Japanese font may seem to be extravagant. However in actual use, one rarely uses all 6000 characters. A statistic says that the most frequent 2000 characters will cover 99% of ordinary Japanese text, so the actual requirements are much less than 33 subfonts.

Once decided on the font configuration, it is straightforward to modify TEX. TEX's input mouth is extended to eat Japanese characters and send an appropriate (subfont, character number) pair to its stomach. After that, TEX doesn't notice that it is actually handling Japanese characters!

As for the preparation of Japanese fonts, I didn't use METAFONT. Considering the amount of effort Knuth has spent to generate Computer Modern Typefaces, it would be a five- to ten-year project to devise a good METAFONT definition for all the Japanese characters. Although it will be necessary in the future, I am just content with available dot fonts for the time being and generated necessary font files from them directly.

Japanese character set

Before explaining the division of the Japanese character set into subfonts, it is necessary to explain what we have first. JIS (Japanese Industry Standard) C-6226 defines a "Code of the Japanese Graphic Character Set for Information Interchange". Here in Japan, we usually use this code (referred to as "JIS code" for short) to represent Japanese characters. It contains 6877 characters in total and uses two 7-bit bytes to represent a single character. These two bytes are called "ku" and "ten" in Japanese or simply "first byte" and "second byte". These bytes are taken from the non-control character part of the ASCII character set. Thus you can use ASCII control characters such as Tab, Carriage Return and Line Feed within a sequence of two-byte codes. See the table in Appendix 2 (This table is typeset by JTEX). In this 94 by 94 table, each Japanese character is positioned at the intersection point of first byte row and second byte column. Each byte is represented by corresponding ASCII character in the outermost column and row. Hexadecimal representation of each byte and "ku", "ten" numbers are added for convenience. All characters are grouped into natural categories as follows (we use the word "ku" to refer to a set of characters having the same first byte):

1-ku & 2-ku	symbols
3-ku	numerals & roman alphabets
4-ku	hiragana (phonetic symbols)
5-ku	katakana (phonetic symbols used to represent foreign words)
6-ku	greek alphabets
7-ku	russian alphabets
8-ku	line segments
16,...,47-ku	2965 first level kanji ordered according to their representative reading
48,...,84-ku	3388 second level kanji ordered by radicals and number of strokes

Here the separation of the kanji set into two levels is very important. The first level contains most frequently used kanji while the second level kanji are rarely used. A normal Japanese sentence consists of kanji, hiragana, katakana and some punctuation symbols. But you can freely mix foreign alphabets within a Japanese sentence, and we do write such a mixture from time to time, so various foreign alphabets are also included in this table.

There are several ways to represent a file with both ASCII and JIS characters in it. If your machine uses an 8-bit byte to represent an ASCII character, simply turning the most significant bit on for all two-byte codes enables you to distinguish ASCII and JIS code easily. This is used in VAX Kanji Code. Some Japanese word processors use so called Shift JIS code which also uses two 8-bit bytes. However the most widely used internal representation is to use escape sequences. JIS codes are simply represented by a sequence of two 7-bit bytes and a sequence of them are sandwiched between three-byte escape sequences ("<esc>$@" or "<esc>$B" to start and "<esc>(J" or "<esc>(B" to end.)

These various formats are easily interchangeable. So JTEX assumes that its input file is a sequence of 7-bit ASCII codes with JIS code parts surrounded by escape sequences.

Division into subfonts

The Japanese character set described in the previous section is divided into the following 33 subfonts. This division naturally corresponds to the categories mentioned above. The control sequence name for each subfont is used to refer to the individual characters in each subfont. Usually a user is not aware of the existence of subfonts, but if he wishes, he can specify, say, the second character in 4-ku, by "{\jhira\char2}".

\jsy	1-ku & 2-ku (symbols)
\jroma	3-ku (numerals & roman alphabets)
\jhira	4-ku (hiragana)
\jkata	5-ku (katakana)
\jgreek	6-ku (greek alphabets)
\jrussian	7-ku (russian alphabets)
\jkeisen	8-ku (line segments)
\ja,...,\jl	16-ku,...,47-ku (first level kanji)
\jm,...,\jz	48-ku,...,84-ku (second level kanji)

In each subfont, a character code corresponds to "ten" number except in kanji subfonts. 26 kanji subfonts (\ja,\jb,...,\jz) all have 256 kanji characters in them except \jl and \jz. Kanji in each level are densely packed into 256 character positions of each subfont in their order. So the last subfont in level 1 (\jl) has only 49 ($= 2965 - 256 \times 11$) kanji and the last one in level 2 (\jz) has only 60 ($= 3388 - 256 \times 13$). Appendix 3 shows the font tables for several subfonts generated by JTEX and the ordinary testfont.tex. Note that these control sequences are generic, i.e. these subfont selectors are assigned the actual subfont by a single control sequence defined in jplain.tex (plain file for JTEX, see below).

Font selection

JTEX provides several different fonts for Japanese and jplain.tex defines useful font selectors which switch all the necessary subfonts at once. For example, a default font is selected by a following control sequence (\jstd) in jplain.tex (This definition is simplified a little):

```
\font\djsystd=dnpjsy38
\font\djhirastd=dnpjhira38
\font\djkatastd=dnpjkata38
\font\djastd=dnpjka38
\jfont\djbstd=dnpjkb38 dnpjka38
\jfont\djcstd=dnpjkc38 dnpjka38
...
\jfont\djkstd=dnpjkk38 dnpjka38
\font\djlstd=dnpjkl38
\def\jstd{\let\jsy=\djsystd
\let\jhira=\djhirastd
\let\jkata=\djkatastd
\let\ja=\djastd \let\jb=\djbstd
...
\let\jk=\djkstd \let\jl=\djlstd
\baselineskip=18pt
\jintercharskip=0.0pt plus0.08pt
\jspaceskip=9.1542pt %38dots on 300dpi
\jasciikanjiskip=1.66667pt
 plus0.83333pt minus0.55556pt}
```

Note that only 15 subfonts (corresponding to \jsy, \jhira, \jkata, \ja, ..., \jl) are preloaded and switched by this command. Users must specify each subfont separately if they want to use foreign alphabets or level 2 kanji. However it is much better to use TEX's fonts for roman, greek and even russian alphabets in normal application. So we encourage people to use TEX's fonts instead of JIS foreign alphabets. \jsmall used in the example input file of Appendix 1 is another example of a font selector.

The control sequence \jfont is introduced to save JTEX's memory space for font information. Most kanji subfonts have identical TFM file and the use of this command:

```
\jfont\fontname=fontfile1 fontfile2
```

enables to load fontfile1 as \fontname using the already loaded font information for fontfile2. Thus it does not consume any font space at all.

Modification to TEX's input mouth

Now we can state the task of JTEX's input mouth clearly. Treat every character in the input file as TEX does except for JIS codes surrounded by escape sequences. For those two-byte codes, deceive TEX as if it has seen the corresponding subfont selector and an appropriate \char command. For example, if you have the following line in the input file:

```
...<esc>$@$3$1$OF|K\8l$G$9!#<esc>(J...
```

it should be seen as if they were:

```
...{\jhira\char19\char76\char47}%
{\ji\char111}{\jk\char37}{\jd\char59}%
{\jhira\char39\char25}{\jsy\3}...
```

(Try to decipher it using the font table in Appendix 2.) There is a little lie in this description. JTEX performs two other things to ensure the proper treatment of Japanese text. First, it inserts \jintercharskip between every pair of Japanese characters. Secondly, it inserts \jasciikanjiskip between an ASCII character and a Japanese one.

For the normal setting of these glues in `jplain.tex`, see the excerpt in the previous section.

The first glue ensures that Japanese sentences can break at any point except "Kinsoku Shori" explained below. And if it is necessary, this glue can stretch a bit to enable right justification. The second glue puts an appropriate amount of space between an English word and a Japanese character.

The internal data structure for tokens is also extended, but I do not describe it here.

Kinsoku Shori

Normal Japanese sentences can break at any point as I stated above. But there are exceptions. These exceptions are called "Kinsoku" in Japanese and the proper treatment of "Kinsoku" is "Kinsoku Shori". Certain characters cannot appear at the beginning of a line (such as close parenthesis and comma) and certain other characters (such as open parenthesis) cannot happen at the end of a line. These conditions are naturally met in TEX if you write these characters next to or just before the neighboring character without inserting space. But in JTEX, glues are put into every gap between Japanese characters so you need to get rid of this extra glue between a "Kinsoku" character and its neighbor.

Spacing

Although the number of characters are many, the saving feature of Japanese characters is that they all have the same width and height. There is no kerning, no ligatures, so typesetting is simpler than English in a sense. But there are a few characters you must pay attention to. They are punctuation marks such as period and comma. We have Japanese period "maru" (1-ku 3-ten) and Japanese comma (1-ku 4-ten). For these characters JTEX provides "Japanese space factor code" (`jsfcode`) whose function is similar to that of `sfcode` in TEX. Whenever JTEX encounters a Japanese character, this `jsfcode` is used instead of `sfcode`.

Carriage return is treated a little bit differently in JTEX. Single carriage return in JIS characters is not equivalent to space. So no extra glue is inserted there. But two or more consecutive carriage returns has the same effect of ending a paragraph as in TEX. ASCII space cannot appear among JIS characters but Tab can appear because it is one of the control characters. This Tab is simply dropped by JTEX. To put it in another words, single carriage return, tab or nothing between Japanese characters are all converted to `\jintercharskip` by JTEX except Kinsoku Shori.

JIS space (1-ku 1-ten, two byte code is "!!") is treated as a normal Japanese character (although it is invisible), so it gives you exactly one character-wide space on output.

Generation of font files from dot fonts

For TEX and device drivers to work, we need two kinds of font files: GF files and TFM files. I generated them from dot fonts by a simple LISP program.

For the first few months, I tried to gather as many Japanese fonts in dot format as possible to enhance JTEX's Japanese fonts. There were not many, but I have found two 24-dot fonts and two 32-dot fonts. One of the 24-dot fonts is part of the JIS standard for dot printers and one can freely copy and distribute it. In the beginning, there was no other way, so I mechanically generated 36, 48 and 72-dot fonts from this JIS 24-dot font to satisfy the need for larger fonts.

But recently we started collaboration with DNP (Dai Nippon Printing Co., one of the biggest printing companies in Japan), and they provide us with fonts of various sizes. We found out that a 38-dot font goes well with TEX's standard 10 point font, so we are preparing the fonts (both Mincho style and Gothic style) with the following dot sizes:

	5pt	6pt	7pt	8pt	9pt	10pt	12pt	17pt
1	19	23	27	30	34	38	46	65
$\sqrt{1.2}$	21	25	29	33	37	42	50	71
1.2	23	27	32	36	41	46	55	78
1.2^2	27	33	38	44	49	55	66	93
1.2^3	33	39	46	53	59	66	79	112
1.2^4	39	47	55	63	71	79	95	134
1.2^5	47	57	66	76	85	95	113	161

These are for the 300 dpi printers, so if you change the resolution you need different sizes as well.

Another important factor when generating Japanese fonts from dot fonts is where to draw the baseline. If you put a box surrounding a Japanese character just on the baseline, non-uppercase ASCII characters look sunk under the baseline. It is difficult to find the optimal point, but we experimentally settled on the following solution. Place the baseline one sixth of the box height above the bottom edge of the box.

Modification to device drivers

As I stated earlier, I tried to make the necessary modification as small as possible. But you need some modification to the device drivers if you want to run the whole system efficiently.

For example, we are now using DEC2065 and IMAGEN 8/300 and 3320 printers. A device driver for this combination is known as DVIIMP. This program loads all the information for a font when it first encounters a new font. And this becomes a great overhead if you use it on dvi files generated by JTEX. Japanese kanji are grouped into subfonts only by code order and there is no "working set" property ("use of one character in a font implies the use of other characters in the same font for a while") among them. So I modified this device driver to load the font information of each character one by one.

On UNIX machines (we use SUN-2 and SUN-3), there is no device driver which directly uses the GF file and most of them use old PXL files. So Japanese fonts are converted to extended PXL format (extended because it contains more than 128 characters, but the basic structure is the same), and device drivers are modified to accept this extended PXL formats.

I made a similar modification to the previewer in X-window system (xdvi) and it is running on our SUNs.

Restrictions

JTEX is quite general and can be used as widely as TEX itself. But there are still minor restrictions.

- You cannot use Japanese characters directly in math mode. But you can always escape to horizontal mode using `\hbox`, so this is not a real restriction.
- You cannot use Japanese characters in control sequence names. But no one has ever wanted to do that until now.
- The number of Japanese fonts usable in a job is limited. This limit is 17, because one Japanese font preloaded by JTEX normally consists of 15 subfonts and JTEX allows a maximum of 256 fonts. If you use other fonts of TEX or level 2 kanji or JIS foreign alphabets besides normally loaded Japanese fonts, you must be content with fewer Japanese fonts. But for ordinary purposes, this number is just enough. If you try to typeset a really big dictionary, you may reach this limit. But well before that you will face the following restriction.
- The IMAGEN print server we are using allows only 3072 different characters per job, and only 653302 bytes for font information per job. This is a real restriction for Japanese TEX. For example, Appendix 1 (only 4 pages!) cannot be output at once and you need to separate it into individual pages.

A bit of history

There have been several attempts to use TEX to format Japanese text. The first and pioneering work was done by Fujita [1]. It was based on TEX78. The modification of SAIL code together with the improvements on output device was carried out to make a usable system. I heard that it is still running at his lab, but now it is obsolete.

Another one was reported by Nagashima and Kawabata [2] at the second Japanese TEX Users Group meeting. (For information on the Japanese TEX Users Group, see TUGboat vol.7, no.3, p.192.) They preprocess a file containing Japanese text to feed it into TEX's mouth. I got hints from their work, so this preprocess is similar to JTEX's input processing, but they literally converted Japanese characters to font selector and `\char` pair producing an expanded intermediate file. They pointed out various problems: inability to use more than two Japanese fonts (they have been working with PXL font files), incompatibility with TEX's magnification sequence, to name a few.

After hearing their talk, I quickly realized that you can dispense with the preprocessor if you use the macro facility of TEX. So the first version of JTEX was realized as a macro package without changing TEX itself [3]. Although the quality of fonts was not so good in the beginning, people were amazed by the fact that TEX can typeset Japanese with only about 500 lines of macros! I did the first version just to see the feasibility of using TEX for Japanese, but people around us started to use JTEX as a daily tool and lots of them complained about its inefficiency and poor font quality.

To improve the efficiency I internalized what the macro does by modifying TEX itself. The number of changes required is not so great (about 40 change items are added to the change file) and the result is superb! The current version of JTEX processes a file with Japanese as fast as original TEX. Also LATEX, AMS-TEX etc. have been extended just by preloading `lplain.tex` and `amstex.tex` into JTEX.

To improve the font quality, we have just started collaboration with DNP (Dai Nippon Printing Co.). They have their own high quality font in

vector format, and they kindly provide us with dot format versions of various sizes. An example output in Appendix 1 and sample font tables in Appendix 3 use these fonts from DNP.

Availability

JTEX is public domain software. It now works on DEC2065 under TOPS-20 version 6.1 and on various UNIX 4.2bsd systems. Several universities have began to use it on their UNIX machines. If you have a running TEX with a decent device driver, JTEX should work too without trouble. (You may need to modify your driver a bit.)

Font files generated from the JIS 24-dot font are also public and can be obtained from the author.

Several modified utility programs such as a previewer are also available.

Future work

What is necessary for using TEX to typeset Japanese is almost completed with JTEX. But there remain many things to be done if you consider JTEX as a total typesetting system.

- Some Japanese texts are written up to down. And we need to support that. But this is rather simple. Just rotate the font 90 degrees counterclockwise and adjust the centerline of each character if necessary.
- We need to build a collection of macro packages to facilitate the use of JTEX in various applications. Locally various forms are converted to JTEX format, and several Japanese academic societies show interest in using JTEX for the publication of their journals. You may see the emergence of JMS-JTEX in the near future....
- It may be necessary to enlarge the number of Japanese fonts usable in one document. This is not so difficult. I could have done so if I wished, and I am ready to do so if there are sufficient demands.
- Enhancement of Japanese fonts is really needed. To define all Japanese characters in METAFONT is a great challenge. And in the long run, someone or a group of people, preferably consisting of both font designers and computer scientists, must do it.

Acknowledgement

My thanks go to many people. Mr. Enari and Mr. Ishii of DNP for providing us with the good dot fonts in various sizes, Prof. Samuel for helping me to modify DVIIMP and giving me useful information on IMAGEN printers. Mr. Amamiya and Mr. Goto who provided the machine environment in which I can work, and many colleagues who used and criticized and found bugs in early versions of JTEX.

Finally I want to mention that without "TEX: The program" my endeavor to extend TEX to Japanese typesetting would have taken more time than I could afford. It has been both fun and exciting to read through the book seeking the best solution for the problem. So my thanks also go to Prof. Knuth.

References

[1] Hiroshi FUJITA: "Technical document typesetting system: TEX" (in Japanese), Information Processing, vol.25, no.8, pp.848–853 (Aug. 1984).

[2] Masaaki NAGASHIMA and Youichi KAWABATA: "Printing Japanese language using TEX" (in Japanese), handout of the 2nd Japanese TEX Users Group meeting (Jul. 1986).

[3] Yasuki SAITO: "Japanese TEX" (in Japanese), Working Group on Japanese Document Processing 10-3, IPJSS (Jan. 1987).

Appendix 1 Sample Input file and Output generted by jTEX

```
% 高木貞治　解析概論　改訂第三版　101ページより抜粋 %
\font\ninerm=cmr9 \font\ninei=cmmi9 \font\ninesy=cmsy9
\nopagenumbers \parindent=\jspaceskip
\def\small{\textfont0=\ninerm \scriptfont0=\sevenrm \textfont1=\ninei
\scriptfont1=\seveni \textfont2=\ninesy \scriptfont2=\sevensy}

定理35. $f(x)$が積分区間内の一点において連続ならば，その点において積分函数$F(x)$は微分可能で
$$ F'(x)=f(x). $$
[証] まず$h>0$として，前のように
$$ F(x+h)-F(x)=\int_{x}^{x+h}f(t)dt. $$
\noindent従って
$$ m \leq {{F(x+h)-F(x)}\over h} \leq M,\qquad\hbox{(\S31, ($7^\circ$)),}$$
\noindent$M, m$は$[x,x+h]$における$f(x)$の値の上限，下限である． 従って
$$ \biggl| {{F(x+h)-F(x)}\over h} - f(x) \biggr| \leq M-m. $$
\noindent さて$f(x)$の連続性によって，$\varepsilon>0$に対して$\delta$を十分小さく
取って，$h<\delta$のとき$M-f(x)<\varepsilon, f(x)-m<\varepsilon,$
従って$M-m<2\varepsilon$ならしめることができる．\par
$h<0$としても同様であるから$F'(x)=f(x)$.

{\jsmall\ninerm\parindent=1.5cm
\item{[注意1]} $\small x$において$\small f(x)$が右へ，あるいは左へ，連続ならば
$\small D^{+}F(x)=f(x)$あるいは$\small D^{-}F(x)=f(x)$.
\item{[注意2]} 同じ条件の下において，積分$\small \int_{x}^{b}f(x)dx$を下の限界に
関して微分すれば$\small -f(x)$を得る．それは$\small \int_{x}^{a}=-\int_{a}^{x}$だから，当
然である．}

逆に$f(x)$が連続で，その一つの原始函数$F(x)$が知られるときは，それを用いて
$f(x)$の積分が計算される．すなわちその場合，かりに
$$ F_1(x)=\int_{a}^{x} f(x)dx $$
\noindent と書けば，${F_1}'(x)=f(x)$, $F'(x)-{F_1}'(x)=0$. 故に$F(x)-F_1(x)=C$は定数である．
すなわち$$ \int_{a}^{x}f(x)dx=F(x)+C.$$
\noindent ここで$x=a$とすれば，$0=F(a)+C$, 故に$C=-F(a)$, 従って，$x=b$とすれば
$$ \int_{a}^{b}f(x)dx=F(b)-F(a). \eqno(1) $$
これを微分積分法の基本公式という．

{\jsmall\ninerm
積分$\small \int_{a}^{x}f(x)dx$の上の限界を変数とし，下の限界を任意の定数とすれば，
その定数をどうきめても，差は$\small x$に無関係である．すなわち$\small f(x)$が積分可能なる区間
に属する任意の定数$\small a, a'$に関して
$\small \int_{a'}^{x}=\int_{a}^{x}-\int_{a}^{a'}$
で，$\small \int_{a}^{a'}$は$x$に関係しない．このように積分の下の限界なる定数を
指定しない場合に，積分を限界なしに$\small \int f(x)dx$と書いて，それを不定積分という．
$\small f(x)$が連続函数ならば，不定積分は原始函数と同意語である．

基本公式(1)は，要約すれば連続函数に関する限り，微分と積分とが互いに逆な算法で
あることを意味する．もしも連続性を仮定しないならば，この関係は成立しない．
すなわち$\small F'(x)=f(x)$でも$\small f(x)$は必ずしも連続でなく，従って必ずしも積分可能でな
いが，また積分可能でも積分函数は$\small F(x)$と合致するとはいわれない．
$\small \int_{a}^{x}f(x)dx$は必ず連続であるけれども，それは必ずしも微分可能でなく，微分可能で
も微分商は$\small f(x)$と合致するとは限らない．連続函数以外では、微分積分法はむずかしい！
\vfill\eject}
\end
```

定理35. $f(x)$ が積分区間内の一点において連続ならば，その点において積分函数 $F(x)$ は微分可能で

$$F'(x) = f(x).$$

[証] まず $h > 0$ として，前のように

$$F(x+h) - F(x) = \int_x^{x+h} f(t)dt.$$

従って

$$m \leq \frac{F(x+h) - F(x)}{h} \leq M, \qquad (\S 31, (7^\circ)),$$

M, m は $[x, x+h]$ における $f(x)$ の値の上限，下限である．従って

$$\left| \frac{F(x+h) - F(x)}{h} - f(x) \right| \leq M - m.$$

さて $f(x)$ の連続性によって，$\varepsilon > 0$ に対して δ を十分小さく取って，$h < \delta$ のとき $M - f(x) < \varepsilon, f(x) - m < \varepsilon$, 従って $M - m < 2\varepsilon$ ならしめることができる．

$h < 0$ としても同様であるから $F'(x) = f(x)$.

[注意1] x において $f(x)$ が右へ，あるいは左へ，連続ならば $D^+F(x) = f(x)$ あるいは $D^-F(x) = f(x)$.

[注意2] 同じ条件の下において，積分 $\int_x^b f(x)dx$ を下の限界に関して微分すれば $-f(x)$ を得る．それは $\int_x^a = -\int_a^x$ だから，当然である．

逆に $f(x)$ が連続で，その一つの原始函数 $F(x)$ が知られるときは，それを用いて $f(x)$ の積分が計算される．すなわちその場合，かりに

$$F_1(x) = \int_a^x f(x)dx$$

と書けば，${F_1}'(x) = f(x), F'(x) - {F_1}'(x) = 0$. 故に $F(x) - F_1(x) = C$ は定数である．すなわち

$$\int_a^x f(x)dx = F(x) + C.$$

ここで $x = a$ とすれば，$0 = F(a) + C$, 故に $C = -F(a)$, 従って，$x = b$ とすれば

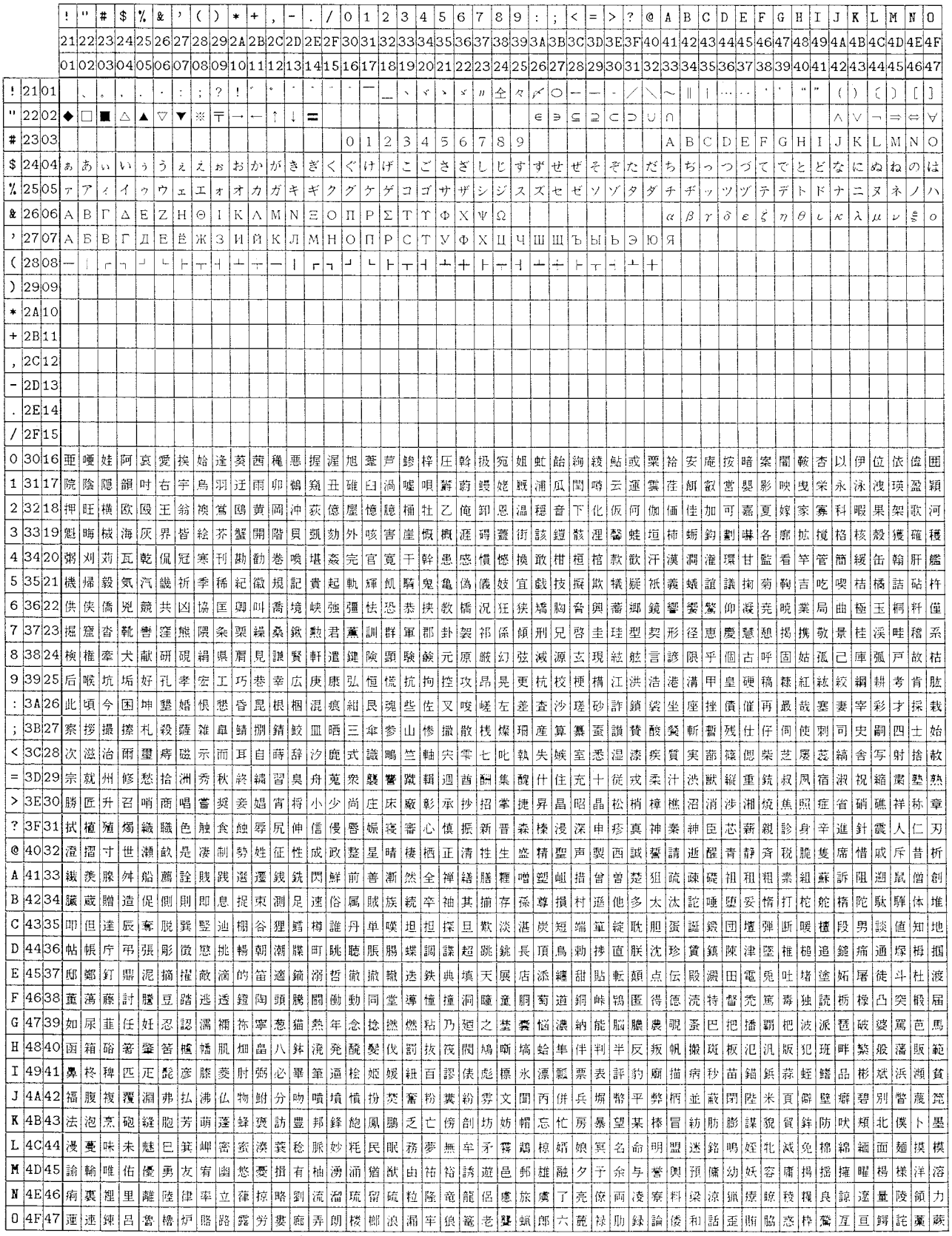
$$\int_a^b f(x)dx = F(b) - F(a). \tag{1}$$

これを微分積分法の基本公式という．

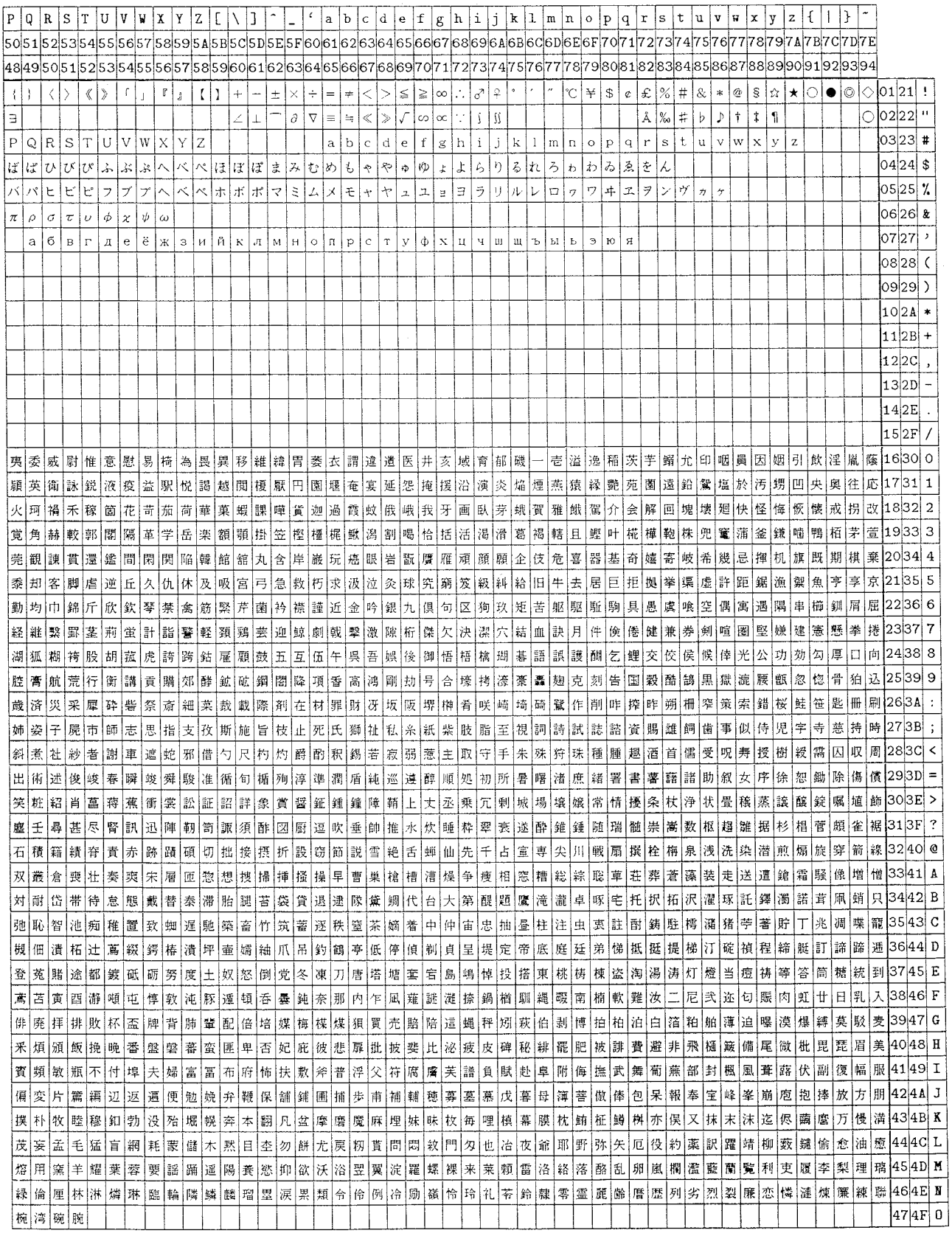
積分 $\int_a^x f(x)dx$ の上の限界を変数とし，下の限界を任意の定数とすれば，その定数をどうきめても，差は x に無関係である．すなわち $f(x)$ が積分可能なる区間に属する任意の定数 a, a' に関して $\int_{a'}^x = \int_a^x - \int_a^{a'}$ で，$\int_a^{a'}$ は x に関係しない．このように積分の下の限界なる定数を指定しない場合に，積分を限界なしに $\int f(x)dx$ と書いて，それを不定積分という．$f(x)$ が連続函数ならば，不定積分は原始函数と同意語である．

基本公式 (1) は，要約すれば連続函数に関する限り，微分と積分とが互いに逆な算法であることを意味する．もしも連続性を仮定しないならば，この関係は成立しない．すなわち $F'(x) = f(x)$ でも $f(x)$ は必ずしも連続でなく，従って必ずしも積分可能でないが，また積分可能でも積分函数は $F(x)$ と合致するとはいわれない．$\int_a^x f(x)dx$ は必ず連続であるけれども，それは必ずしも微分可能でなく，微分可能でも微分商は $f(x)$ と合致するとは限らない．連続函数以外では、微分積分法はむずかしい！

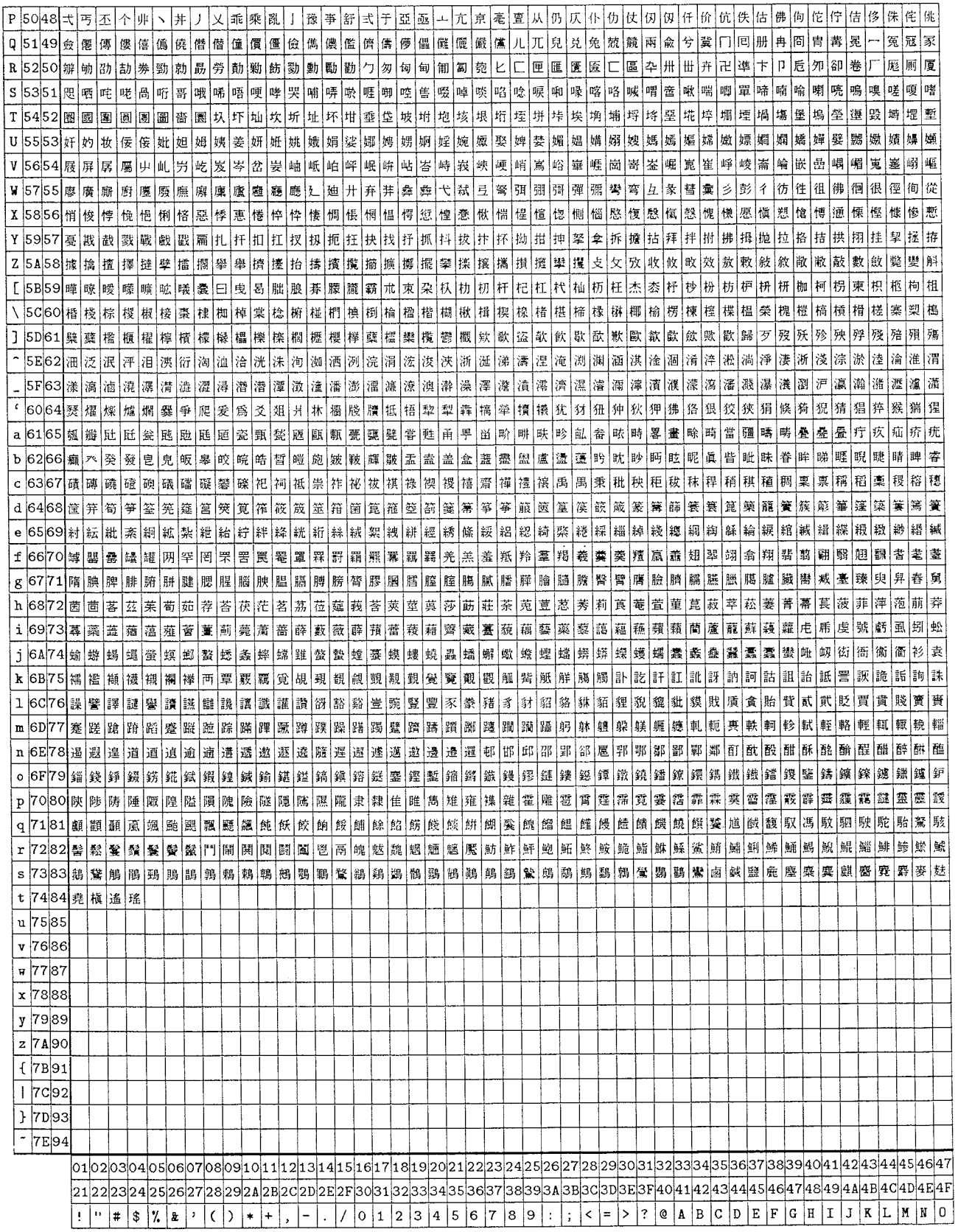
Appendix 2 JIS Code Table (top left quarter)



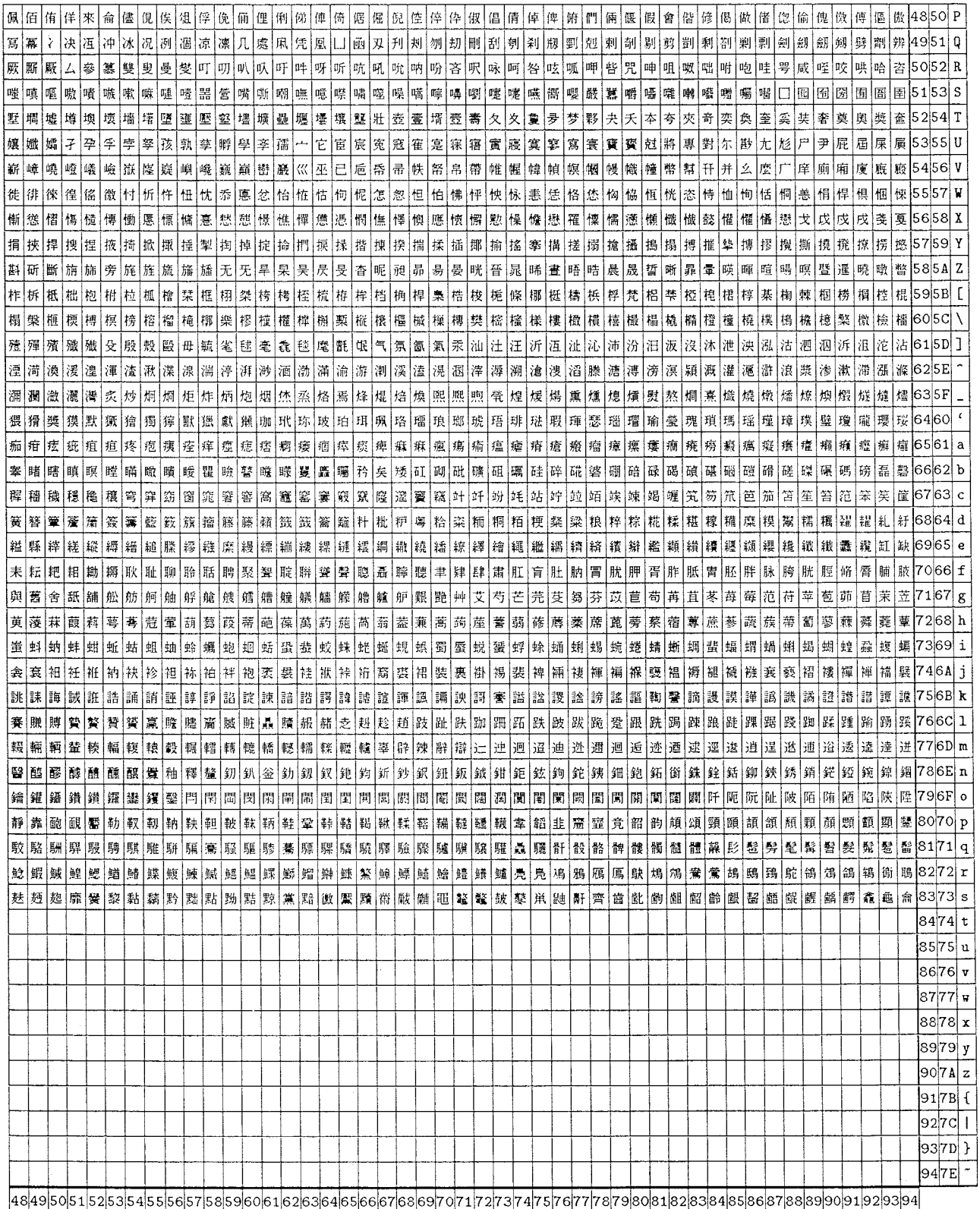
JIS Code Table (top right quarter)



JIS Code Table (bottom left quarter)



JIS Code Table (bottom right quarter)



Appendix 3 Sample Font Table for several subfonts

Test of dnpjhira38 on May 16, 1987 at 1254

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x		ぁ	あ	ぃ	い	ぅ	う	ぇ	"0x
'01x	え	ぉ	お	か	が	き	ぎ	く	
'02x	ぐ	け	げ	こ	ご	さ	ざ	し	"1x
'03x	じ	す	ず	せ	ぜ	そ	ぞ	た	
'04x	だ	ち	ぢ	っ	つ	づ	て	で	"2x
'05x	と	ど	な	に	ぬ	ね	の	は	
'06x	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	"3x
'07x	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み	
'10x	む	め	も	ゃ	や	ゅ	ゆ	ょ	"4x
'11x	よ	ら	り	る	れ	ろ	ゎ	わ	
'12x	ゐ	ゑ	を	ん					"5x
'13x									
	"8	"9	"A	"B	"C	"D	"E	"F	

Test of dnpjkata38 on May 16, 1987 at 1254

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x		ァ	ア	ィ	イ	ゥ	ウ	ェ	"0x
'01x	エ	ォ	オ	カ	ガ	キ	ギ	ク	
'02x	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	"1x
'03x	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	
'04x	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	"2x
'05x	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	
'06x	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	"3x
'07x	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ	
'10x	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	"4x
'11x	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	
'12x	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ		"5x
'13x									
	"8	"9	"A	"B	"C	"D	"E	"F	

Test of dnpjka38 on May 16, 1987 at 1254

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	亜	啞	娃	阿	哀	愛	挨	姶	"0x
'01x	逢	葵	茜	穐	悪	握	渥	旭	
'02x	葦	芦	鯵	梓	圧	斡	扱	宛	"1x
'03x	姐	虻	飴	絢	綾	鮎	或	粟	
'04x	袷	安	庵	按	暗	案	闇	鞍	"2x
'05x	杏	以	伊	位	依	偉	囲	夷	
'06x	委	威	尉	惟	意	慰	易	椅	"3x
'07x	為	畏	異	移	維	緯	胃	萎	
'10x	衣	謂	違	遺	医	井	亥	域	"4x
'11x	育	郁	磯	一	壱	溢	逸	稲	
'12x	茨	芋	鰯	允	印	咽	員	因	"5x
'13x	姻	引	飲	淫	胤	蔭	院	陰	
'14x	隠	韻	吋	右	宇	烏	羽	迂	"6x
'15x	雨	卯	鵜	窺	丑	碓	臼	渦	
'16x	嘘	唄	鬱	蔚	鰻	姥	厩	浦	"7x
'17x	瓜	閏	噂	云	運	雲	荏	餌	
'20x	叡	営	嬰	影	映	曳	栄	永	"8x
'21x	泳	洩	瑛	盈	穎	頴	英	衛	
'22x	詠	鋭	液	疫	益	駅	悦	謁	"9x
'23x	越	閲	榎	厭	円	園	堰	奄	
'24x	宴	延	怨	掩	援	沿	演	炎	"Ax
'25x	焔	煙	燕	猿	縁	艶	苑	薗	
'26x	遠	鉛	鴛	塩	於	汚	甥	凹	"Bx
'27x	央	奥	往	応	押	旺	横	欧	
'30x	殴	王	翁	襖	鴬	鴎	黄	岡	"Cx
'31x	沖	荻	億	屋	憶	臆	桶	牡	
'32x	乙	俺	卸	恩	温	穏	音	下	"Dx
'33x	化	仮	何	伽	価	佳	加	可	
'34x	嘉	夏	嫁	家	寡	科	暇	果	"Ex
'35x	架	歌	河	火	珂	禍	禾	稼	
'36x	箇	花	苛	茄	荷	華	菓	蝦	"Fx
'37x	課	嘩	貨	迦	過	霞	蚊	俄	
	"8	"9	"A	"B	"C	"D	"E	"F	