Special-Purpose OpenType Japanese Font Tutorial: Kazuraki

1.1 Introduction

This tutorial is designed to guide Japanese font developers in building special-purpose OpenType® Japanese fonts, using KazurakiSP2N-Light (to be referred to as simply *Kazuraki*® from this point forward) as an example of how to build a genuinely and completely proportional Japanese font. The techniques, tools, and control files that are described or referenced in, or attached to, this document are tightly coupled to tools that are included in AFDKO (Adobe® Font Development Kit for OpenType) Version 2.0 or greater, which is available, at no charge, at the following URL:

http://www.adobe.com/devnet/opentype/afdko/

Please pay close attention to the last section of this document, which includes information—relevant as of this writing—about compatibility with applications.

If you have any questions regarding the content of this document, please do not hesitate to contact its author, Ken Lunde (*lunde@adobe.com*).

1.2 Kazuraki Design Motivations

With the exception of the vertical-only hiragana ligatures, all glyphs in Kazuraki have corresponding horizontal and vertical forms. That is, for each character, there are two glyphs in the font. The glyphs themselves are the same, but their advances and default positioning along both X- and Y-axes are different. All glyphs needed to be replicated for vertical use, due to limitations in the ability to shift glyphs in both X- and Y-axis directions in the OpenType 'vmtx' table, coupled with the strong desire to make expected behavior the default—without an application depending on, or activating, any GSUB or GPOS features.

The glyph complement includes glyphs for all standard kana characters, but includes a limited set of kanji, 1,483 to be exact, suitable for creating Japanese greeting cards, menus, and other specialized uses. Kazuraki also includes a basic set of proportional Latin glyphs to aid in keyboard input. At a minimum, we recommend including glyphs that correspond to ASCII (U+0020 through U+007E). While Kazuraki is fully-functional in this limited context, it also serves as an example for building comparable fonts.

1.2.1 Genuine Proportional Glyphs

All horizontal/vertical glyph pairs in Kazuraki are the same, with the exception of small kana, some punctuation, and parenthetical symbols, which require different glyphs for horizontal and vertical use in conventional Japanese fonts, due to their position or orientation. Post-processing of the glyph data is used to derive the vertical versions, which are positioned along the Y-axis differently, and have different metrics.

1.2.2 Vertical-Only Hiragana Ligatures

Kazuraki includes a small number of vertical-only hiragana ligatures, fifty-one to be exact. Three are four-character ligatures, twelve are three-character ligatures, and the remaining thirty-six are two-character ones. These are activated through the use of the 'liga' GSUB feature. The advantage of using 'liga' is that most applications that perform an adequate level of typesetting also tend to automatically invoke this GSUB feature,

or at least allow users to activate it through a standard UI. This allows vertical-only hiragana ligatures to be used by default in many applications.

1.2.3 CID- Versus Name-Keyed Structure

Kazuraki was built as a CID-keyed OpenType font. Its 'CFF' table is built from a CIDFont resource. Although Kazuraki could have been built as a name-keyed font, CID-keyed fonts have advantages for Japanese fonts. The primary advantage is that a CID-keyed structure supports multiple hint dictionaries. Each hint dictionary ideally covers glyphs for specific glyph classes, and each hint dictionary can have its own hinting parameters. Using multiple hint dictionaries thus offers significant rendering advantages.

1.3 OpenType Table Settings & Overrides

Many of the OpenType tables require special settings for Kazuraki. This section describes the special settings, one table at a time, along with information that demonstrates how the table overrides are specified in the "features" file as used as input for AFDKO's *makeotf* tool.

1.3.1 BASE

There is no special treatment necessary for the 'BASE' table, other than the usual ICF (Ideographic Character Face) and baseline values that should be normally specified. It is very important to include a 'BASE' table in all OpenType fonts. The following is the 'BASE' table override in the "features" file of Kazuraki:

```
table BASE {
  HorizAxis.BaseTagList              icfb icft ideo romn;
  HorizAxis.BaseScriptList DFLT ideo -117 877  -120 0,
                           hani ideo -117 877  -120 0,
                           kana ideo -117 877  -120 0,
                           latn ideo -117 877  -120 0;
  VertAxis.BaseTagList               icfb icft ideo romn;
  VertAxis.BaseScriptList  DFLT ideo 3    997  0    120,
                           hani ideo 3    997  0    120,
                           kana ideo 3    997  0    120,
                           latn ideo 3    997  0    120;
} BASE;
```

Note that only the 'DFLT', 'hani', 'kana', and 'latn' scripts are declared in the 'BASE' table override, based on the glyph complement of Kazuraki.

1.3.2 CFF

The original source data for Kazuraki is a name-keyed OpenType font containing exactly 1,827 glyphs, each with 1000-unit set widths. The same source font also contains 'palt' and 'vpal' GPOS features that specify the desired glyph metrics (horizontal and vertical set widths, and X- and Y-axis shifting values, respectively), and these GPOS features are used as the source of the default metrics for the horizontal and vertical glyphs in the final font, which is an Adobe- Identity-0 CID-keyed OpenType font containing 3,776 glyphs (CIDs 0 through 3775).

Three AFDKO tools are used to process the original set of 1,827 glyphs: *tx*, *mergeFonts*, and *rotateFont*.

The first tool, *tx*, simply extracts the 'CFF' table from the source OpenType font, and converts it into a name-keyed Type 1 font, using the following command line:

```
% tx -t1 KazurakiSource.otf > font.pfa
```

The second tool, *mergeFonts*, is used to convert the glyph names into CIDs, and to simultaneously synthesize the vertical glyphs from the original horizontal versions, using the following command line:

```
% mergeFonts -cid cidfontinfo cidfont.raw h.map font.pfa v.map font.pfa
```

The end result is an Adobe-Identity-0 CIDFont resource, named "cidfont.raw," containing 3,476 glyphs in a single hint dictionary.

The third tool, *rotateFont*, serves to set the widths for the horizontal glyphs, and to also shift them along the X-axis. The set width and X-axis shifting values are in the 'palt' GPOS feature of the source name-keyed OpenType font. The following command line is used:

```
% rotateFont -t1 -rtf shift.map cidfont.raw cidfont-prop.raw
```

The end result is an Adobe-Identity-0 CIDFont resource, named "cidfont-prop.raw," containing 3,476 glyphs, the horizontal versions of which now have proportional metrics. 300 proportional Latin glyphs are added to bring the glyph complement up to 3,776 glyphs.

As an example, let us explore the treatment of the horizontal glyph for the hiragana character "shi" (し). The glyph in the original name-keyed font is named "CID864" (named after CID+864 of the Adobe-Japan1-*x* character collection). Its 'palt' GPOS feature settings were as follows in the source name-keyed OpenType font:

```
position \CID864 <-223 0 -485 0>;
```

After processing by the *mergeFonts* tool, this (horizontal) glyph becomes CID+224. The 'palt' data shown above is used to generate the following *rotateFont* directive for the "shift.map" mapping file:

```
224 224 515 -223 0
```

The calculation is simple: the value "-223" is used as-is as the X-axis shifting value, and the default set-width value of 1000 becomes 515 after the value "-485" is added to it (a subtraction operation).

The set widths and Y-axis shifting for the vertical glyphs are specified in a 'vmtx' table override definition that is inserted into the "features" file. The handling of vertical glyphs, in terms of specifying their set widths and Y-axis positions, is covered later in this document.

Once these tools have been run, and the glyphs are assigned to CIDs, and the horizontal glyphs have been set to their default set widths and X-axis positions (that is, made proportional), the CIDFont is then hinted as usual. The process of hinting also involves creating multiple hint dictionaries, ideally only one for each glyph class.

NOTE: The process of establishing multiple hint dictionaries in a CIDFont requires files and tools that are not included in AFDKO, and their description is intentionally (and appropriately) omitted from this document. However, mergeFonts *techniques described in Adobe Tech Note #5900 ("AFDKO Version 2.0 Tutorial: mergeFonts, rotateFont & autohint"), which is among the documentation bundled with AFDKO, can be used to establish multiple hint dictionaries. Multiple* mergeFonts *mapping files, in which the first line each each file names a hint dictionary, is the appropriate technique. And, multiple* mergeFonts *mapping files can serve to specify glyphs for single hint dictionaries. In fact, the proprietary tool that was used to establish multiple hint dictionaries for Kazuraki uses* mergeFonts *to perform this task.*

The Special-Purpose Adobe-Identity-0 Character Collection

Because the glyph complement of Kazuraki does not adhere to the Adobe-Japan1-6 character collection, and because it makes little sense to extend Adobe-Japan1-6 to accommodate such special-purpose fonts, the special-purpose Adobe-Identity-0 character collection is advertised in the CFF. Although "Adobe-Identity-0" does not explicitly specify that Kazuraki is a Japanese font, other table settings, along with proven heuristics, are used to make clear the fact that it is a Japanese font.

In essence, the advantage of using the Adobe-Identity-0 character collection is that there are no preconceived notions of language or script, making it possible to build CIDFonts based on dynamic glyph sets, much like TrueType and name-keyed OpenType fonts. The technique of using the Adobe-Identity-0 character collection should not be used to build general-purpose OpenType Japanese fonts. The Adobe-Japan1-*x* character collection should be used instead.

File Size Issues

Because the horizontal/vertical glyph pairs are identical, in terms of their outlines, the subroutinization ability of AFDKO's *makeotf* tool makes the resulting CFF table only slightly larger than that of the original source data, which contained roughly half the number of glyphs. The subroutinized 'CFF' table thus became approximately 50% the size of the unsubroutinized version.

Hinting Issues

Hinting, in terms of stem widths, is applied as usual for Kazuraki. Alignment zones, however, are another matter. The hint dictionaries for non-Latin glyph classes, such as kana and kanji, typically use the following /BlueValues array:

```
/BlueValues [–250 –250 1100 1100] def
```

However, due to the larger (taller) than usual bounding boxes of the vertical-only hiragana ligatures, the hint dictionary for the kana glyphs require different values, in order to ensure that there are no alignment zones in contact with its glyphs. The "Kana" hint dictionary of Kazuraki uses the following /BlueValues array:

```
/BlueValues [–1250 –1250 2000 2000] def
```

Furthermore, the "Dingbats" and "Kanji" hint dictionaries of Kazuraki uses the same /BlueValues array, due to the extent to which the shapes of their glyphs extend above and below the 1000×1000 em-box.

The following is the /FontBBox for Kazuraki:

```
/FontBBox {–338 –1179 1587 1939} def
```

It was thus critical to select /BlueValues values less than –1179 (the Y-axis low point) and greater than 1939 (the Y-axis high point), especially for the "Kana" hint dictionary.

1.3.3 GPOS

The only GPOS features that are included in Kazuraki are 'kern' and 'vkrn', for horizontal and vertical kerning, respectively. The 'palt' and 'vpal' GPOS features in the original source data served to drive the production process, to specify the horizontal/vertical set widths and X- and Y-axis shifting values. These GPOS features are not in the final form of the font, because they are not necessary. Their values were used to define the default glyph metrics.

1.3.4 GSUB

Kazuraki contains only four GSUB features: 'fwid', 'vert', 'vrt2', and 'liga'. Although the conventional ordering of these features is 'liga' followed by 'vert' and 'vrt2', this font's vertical-only hiragana ligatures necessitates a different ordering, specifically 'vert' and 'vrt2' followed by 'liga'. As a general rule, the ordering of GPOS and GSUB features in the "features" file is important, because the same ordering is reflected in the 'GPOS' and 'GSUB' tables that are generated by AFDKO's *makeotf* tool.

The 'vert' GSUB feature substitutes the horizontal forms with their vertical versions. This feature covers the majority of the font. Once the 'vert' GSUB feature has been applied, the vertical-only hiragana ligatures can then be applied via the 'liga' GSUB feature.

OpenType-savvy applications that support vertical writing automatically invoke the 'vert' (or 'vrt2', if present) GSUB feature. These same applications also invoke the 'liga' GSUB feature by default, which then serves to activate (or make default) the vertical-only hiragana ligatures.

1.3.5 OS/2

Because the special-purpose Adobe-Identity-0 character collection is used for Kazuraki, several 'OS/2' table fields must be more carefully specified, such as the OS/2.unicodeRange and OS/2.codePageRange fields. For Kazuraki, these settings are specified in the "features" file as the following 'OS/2' table overrides:

```
XHeight 423;
CapHeight 645;
UnicodeRange 0 1 2 5 31 33 35 36 38 48 49 50 59 62 65 68;
CodePageRange 1252 932;
```

Note that the "XHeight" and "CapHeight" values are set to values that correspond to the proportional Latin glyphs that are in its glyph complement.

The "UnicodeRange" values correspond as follows:

0 Basic Latin
1 Latin-1 Supplement
2 Latin Extended-A
5 Spacing Modifier Letters
31 General Punctuation
33 Currency Symbols
35 Letterlike Symbols
36 Number Forms
38 Mathematical Operators
48 CJK Symbols And Punctuation
49 Hiragana
50 Katakana
59 CJK Unified Ideographs
62 Alphabetic Presentation Forms
65 Vertical Forms
68 Halfwidth And Fullwidth Forms

The "CodePageRange" value of 1252 corresponds to "Latin 1," and 932 corresponds to "JIS/Japan."

These 'OS/2' table settings help to explicitly identify Kazuraki as a Japanese font.

1.3.6 VORG

The 'VORG' table is automatically generated when using AFDKO's *makeotf* tool, and is derived from the settings and overrides of the 'vmtx' table. See the section for the 'vmtx' table for more information on 'vmtx' table settings and overrides.

1.3.7 cmap

The 'cmap' table for CID-keyed OpenType fonts is built using one or more CMap resources. For Kazuraki, because it is based on the special-purpose Adobe-Identity-0 character collection, special-purpose CMap

resources are necessary. Because the vertical glyphs are accessible through the 'vert' and 'vrt2' GSUB features, only the horizontal glyphs are mapped from Unicode code points.

1.3.8 name

The 'name' table is built as usual, setting English and Japanese strings, as appropriate. The only exception is the name.ID=20 string, which is not necessary due to the special-purpose nature of Kazuraki. The specification of 'name' table strings is performed in the "FontMenuNameDB" and "features" files. Care must be taken to explicitly set Japanese as the script and language, for as many of the strings as possible, as appropriate.

For more information about specifying 'name' table strings for OpenType Japanese fonts, please refer to Adobe Tech Note #5149 ("OpenType-CID/CFF CJK Fonts: 'name' Table Tutorial"), available at the following URL:

http://www.adobe.com/devnet/font/pdfs/5149.OTFname_Tutorial.pdf

1.3.9 vmtx

The 'vmtx' table plays an absolutely crucial role in building fonts such as Kazuraki, because it is in this table that the vertical set widths are specified, along with any Y-axis shifting. Anything specified in the 'vmtx' table becomes default behavior. Thus, OpenType-savvy applications that support vertical writing can use such fonts without modification.

Kazuraki's "features" file contains a very large number of "VertAdvanceY" and "VertOriginY" statements in its 'vmtx' table overrides. Nearly every vertical glyph required treatment by one or both of these 'vmtx' overrides.

As an example, let us explore the treatment of the vertical glyph for the hiragana character "shi" (し). The glyph in the original name-keyed font is named "CID864" (named after CID+864 of the Adobe-Japan1-*x* character collection). Its 'vpal' GPOS feature settings were as follows:

```
position \CID864 <0 -26 0 331>;
```

After processing by the *mergeFonts* tool, this (vertical) glyph became CID+2083. The 'vpal' data shown above was used to generate the following 'vmtx' table overrides for the "features" file:

```
VertOriginY \2083 906;
VertAdvanceY \2083 1331;
```

The calculation is simple: the value "-26" is subtracted from 880 (a fixed value that represents the top of the em-box) to become 906, which is the new origin, and the value "331" is added to the default 1000-unit width to become 1331.

1.4 Special Tools

A single special-purpose tool was written, in Perl, to generate all of the control files and data in a single execution. The mapping files that controlled the execution of the *mergeFonts* and *rotateFont* tools were generated by this tool, as was the "features" files containing 'vmtx' table overrides and all GSUB and GPOS feature definitions. The raw data to build the Unicode (UTF-32) CMap resources was also generated by this tool. Due to the large number of glyphs, and the complex relationships between them, it was important to create a tool to do this work, because doing so by hand would have been tedious, and also prone to error.

When writing a comparable tool, I found that it was very useful to maintain a mapping from the glyph names in the source font to the final CIDs. This made generating the raw data for the CMap resource a much easier task. It also made other tasks easier.

1.5 OpenType Control Files & Data

Once the name- to CID-keyed conversion is complete, the usual control files and data, required by AFDKO's *makeotf* tool, must be generated or supplied. These control files and data are detailed in the following sections.

1.5.1 CIDFont Resource

Kazuraki's CIDFont resource is constructed as usual, with an appropriate number of hint dictionaries, ideally one for each glyph class, and with Adobe-Identity-0 as its advertised ROS (/Registry, /Ordering, and /Supplement, which are the three entries of the /CIDSystemInfo dictionary). As stated earlier in this document, Kazuraki's CIDFont resource contains 3,776 glyphs, specifically CIDs 0 through 3775. Kazuraki contains exactly six hint dictionaries, named as follows, and with the number of glyphs in each in parentheses:

- KazurakiSP2N-Light-Dingbats (102 glyphs)
- KazurakiSP2N-Light-Generic (one glyph)
- KazurakiSP2N-Light-Kana (407 glyphs)
- KazurakiSP2N-Light-Kanji (2,966 glyphs)
- KazurakiSP2N-Light-Proportional (150 glyphs)
- KazurakiSP2N-Light-ProportionalRot (150 glyphs)

1.5.2 The "features" File

The "features" file plays an important role, in that overrides to specific tables can be made, and GPOS and GSUB features can be defined. Kazuraki contains two GPOS features, 'kern' and 'vkrn', to specify horizontal and vertical kerning pairs, respectively. Four GSUB features—'fwid', 'vert', 'vrt2', and 'liga'—are also included, whose relative order is important, as described earlier in this document. Lastly, the 'vmtx' table overrides, which are also used to build the 'VORG' table, serve to specify the default vertical metrics.

1.5.3 The "FontMenuNameDB" File

The English and Japanese menu names that are recorded in the 'name' table of an OpenType font are specified in the "FontMenuNameDB" file. Kazuraki's "FontMenuNameDB" entry is shown below:

```
[KazurakiSP2N-Light]
   f=3,1,0x411,\304b\3065\3089\304d SP2N
   s=3,1,0x411,L
   l=3,1,0x411,\304b\3065\3089\304d SP2N L
   f=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N
   s=1,1,11,L
   l=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N L
   f=Kazuraki SP2N
   s=L
   l=Kazuraki SP2N L
```

It is important to stress that English menu names must be set—in addition to the obvious Japanese menu names—in case such fonts are used in applications whose heuristics may cause a failure to properly use the Japanese menu names.

1.5.4 CMap Resources

For special-purpose fonts such as Kazuraki, only a Unicode CMap resource is necessary. Even if there are no mappings outside the BMP, a UTF-32 CMap resource is still recommended as input to AFDKO's makeotf tool. For Kazuraki, the Unicode CMap resource was named "UniKazurakiSP2N-UTF32-H" to make it tightly coupled with the font. This CMap resource is used solely as input to AFDKO's makeotf tool, to build the Unicode 'cmap' subtables of the resulting OpenType font.

For more information about building CMap resources, please refer to Adobe Tech Note #5099 ("Building CMap Files for CID-Keyed Fonts"), available at the following URL:

http://www.adobe.com/devnet/font/pdfs/5099.CMapFiles.pdf

1.6 Testing & Compatibility Considerations

Kazuraki works as expected in Adobe InDesign® CS2 and greater. The horizontal and vertical metrics are respected, and proper vertical layout is supported, including the vertical-only hiragana ligatures.

Kazuraki functions in Adobe Illustrator® CS2 and Adobe Photoshop® CS2 with some limitations, specifically that the vertical-only hiragana ligatures do not function, even if the 'liga' GSUB feature is turned on.

In addition, these and other applications may not display Kazuraki's name in Japanese in their font menus. Kazuraki's name may instead display in English, using the English-language menu name strings that are specified in the 'name' table.

Kazuraki works very well with CS3 and CS4 applications, specifically InDesign, Illustrator, and Photoshop. In fact, we recommend that CS3 and later applications be used for Kazuraki and comparable fonts.

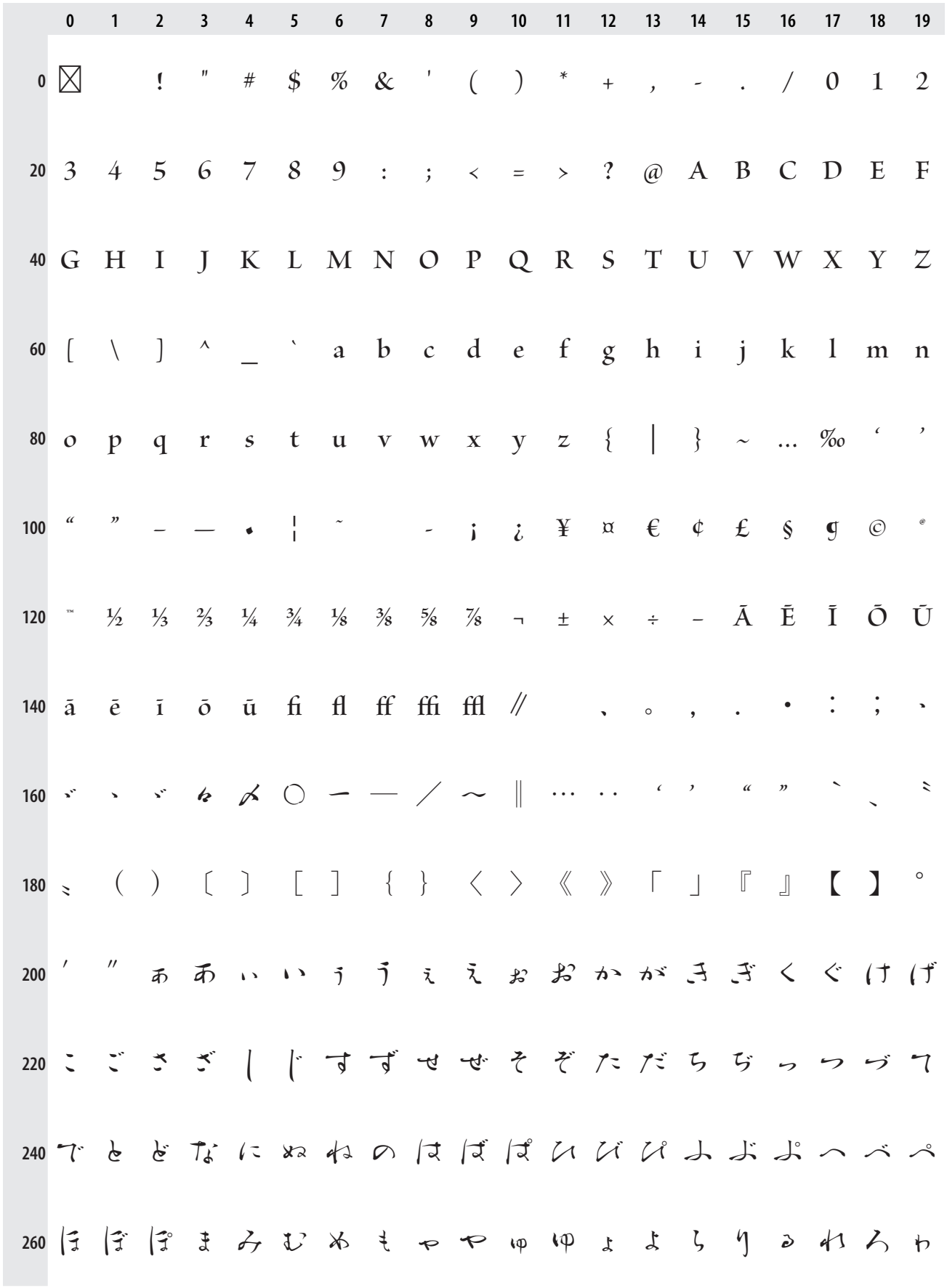
Due to its unique (and limited) glyph complement, Kazuraki is not recommended for use as a component in these applications' Composite Font functionality.

When developing special-purpose OpenType Japanese fonts, it is prudent to rigorously test the font with a variety of OSes and applications, to include entire document authoring workflows.

1.7 Glyph Synopsis

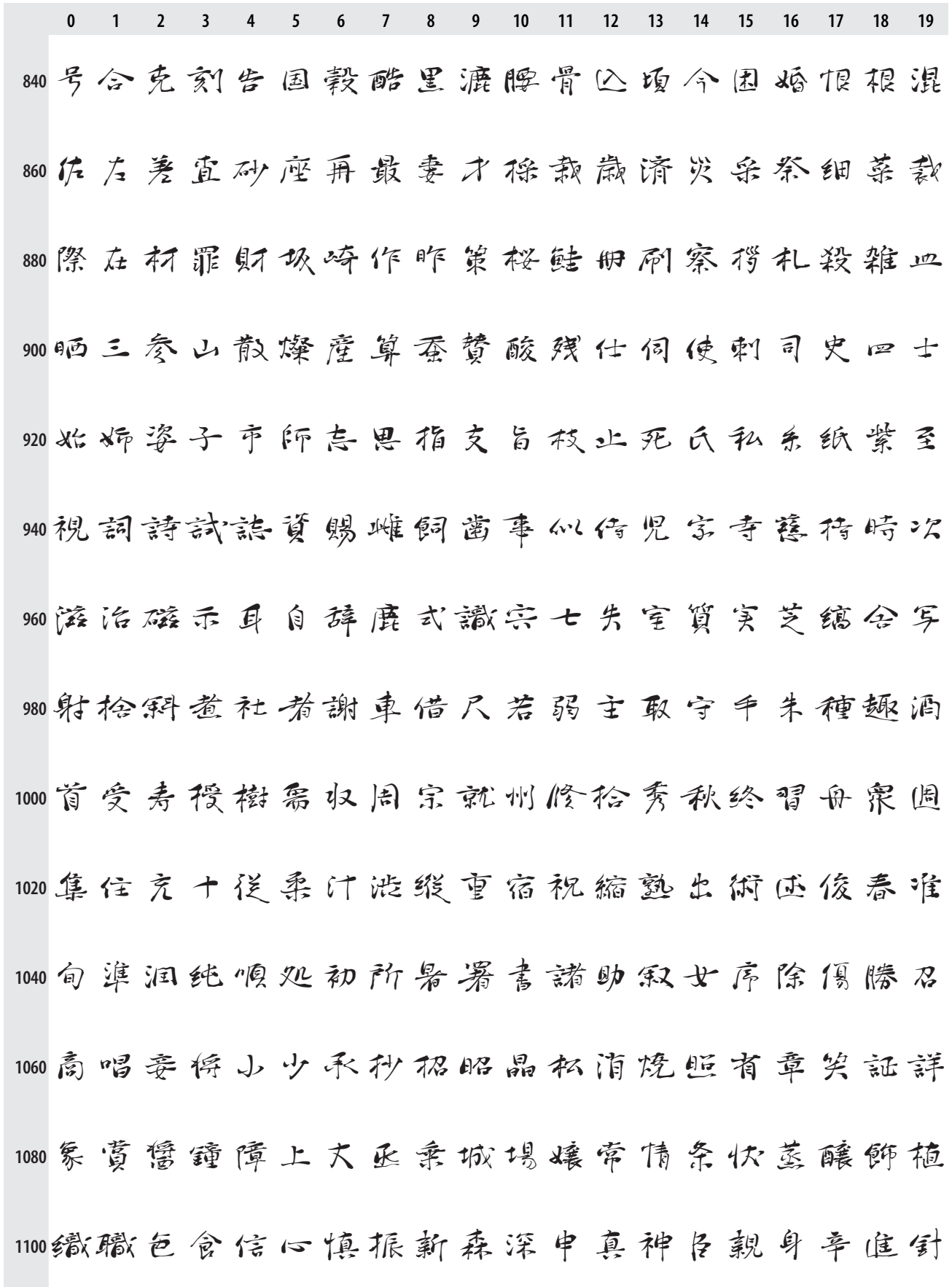
The following fourteen pages provide a complete glyph synopsis for the 3,776 glyphs of Kazuraki, arranged by CID. The following list provides information about specific CID ranges:

1–150	Horizontal glyphs—proportional Latin
151–1863	Horizontal glyphs—Japanese
1864–2013	Pre-rotated forms of CIDs 1–150
2014–3722	Vertical forms of CIDs 151–1863
3723–3775	Vertical-only hiragana ligatures and pre-composed double kana iteration marks

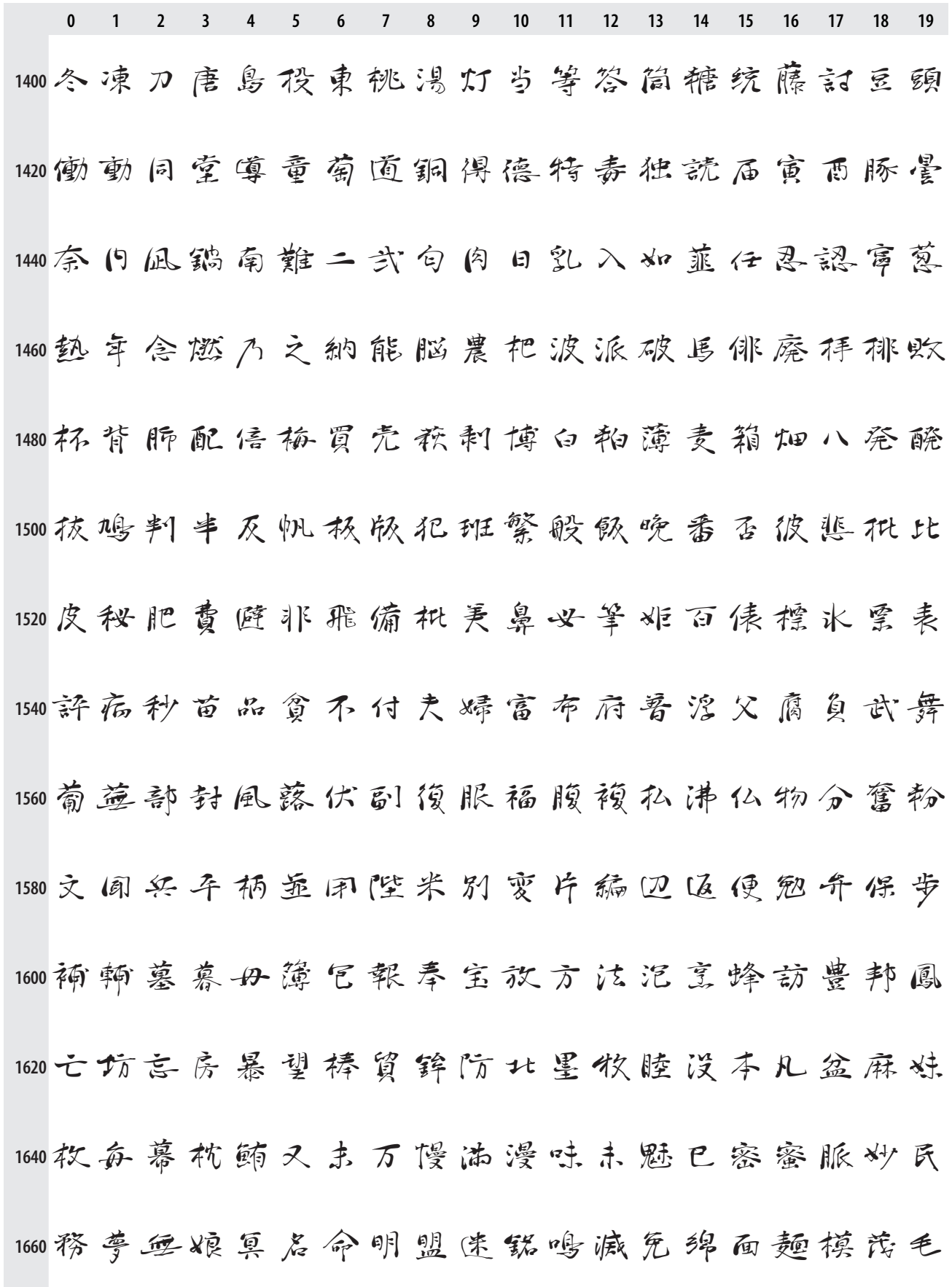

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 ☒ ! " # $ % & ' ( ) * + , - . / 0 1 2
20 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F
40 G H I J K L M N O P Q R S T U V W X Y Z
60 [\ ] ^ _ ` a b c d e f g h i j k l m n
80 o p q r s t u v w x y z { | } ~ … ‰ ‘ ’
100 “ ” – — • ¦ ˜ - ¡ ¿ ¥ ¤ € ¢ £ § ¶ © ®
120 ™ ½ ⅓ ⅔ ¼ ¾ ⅛ ⅜ ⅝ ⅞ ¬ ± × ÷ – Ā Ē Ī Ō Ū
140 ā ē ī ō ū fi fl ff ffi ffl // 、 。 , . • : ; 、
160 〃 〉 〃 〄 々 〇 ー — ／ 〜 ‖ … ‥ ‘ ’ “ ” 〴 〵 〻
180 〻 ( ) 〔 〕 [ ] { } 〈 〉 《 》 「 」 『 』 【 】 °
200 ′ ″ ぁ あ ぃ い ぅ う ぇ え ぉ お か が き ぎ く ぐ け げ
220 こ ご さ ざ し じ す ず せ ぜ そ ぞ た だ ち ぢ っ つ づ て
240 で と ど な に ぬ ね の は ば ぱ ひ び ぴ ふ ぶ ぷ へ べ ぺ
260 ほ ぼ ぽ ま み む め も ゃ や ゅ ゆ ょ よ ら り る れ ろ ゎ

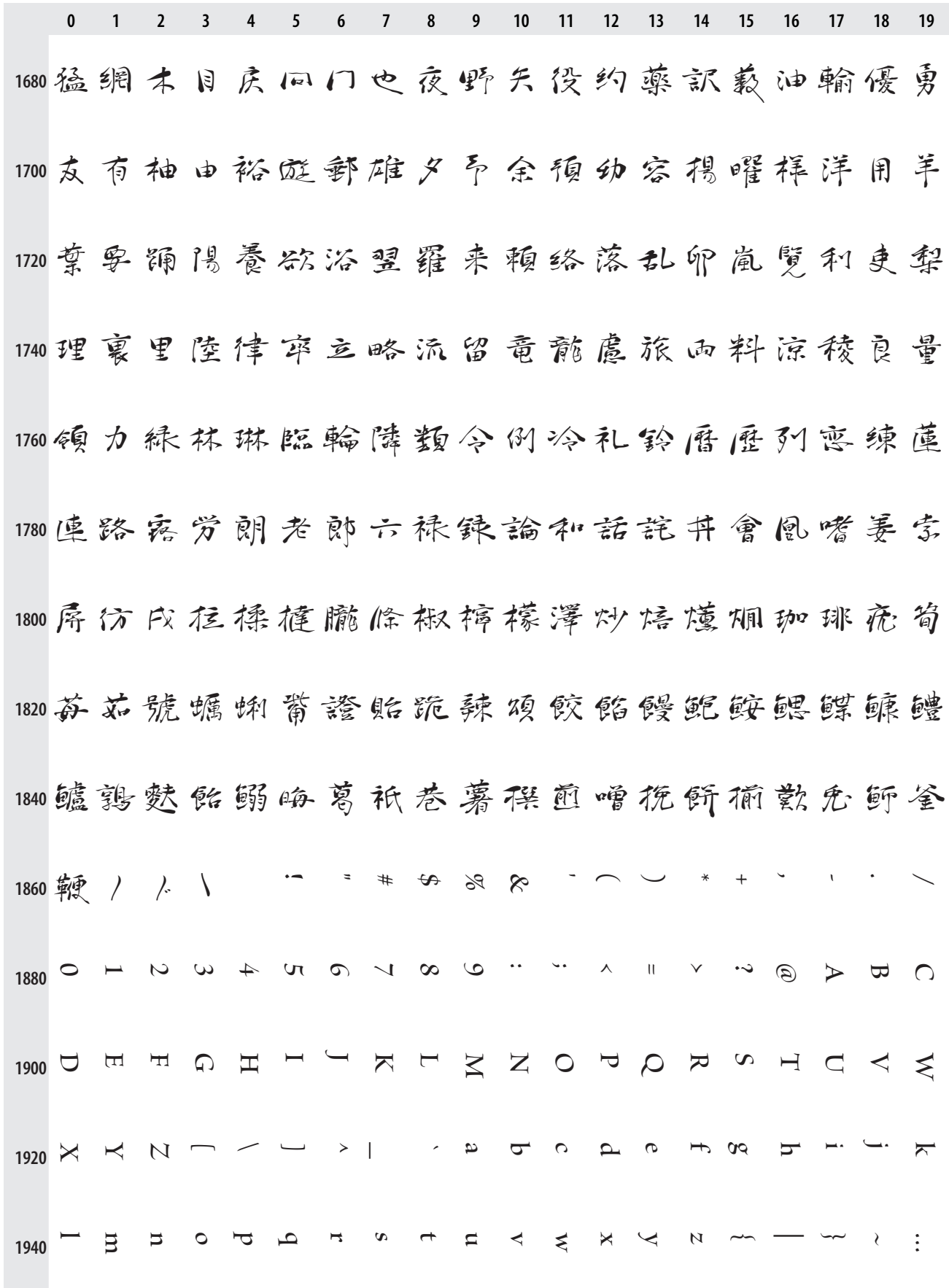

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
280
300
320
340
360
380
400
420
440
460
480
500
520
540

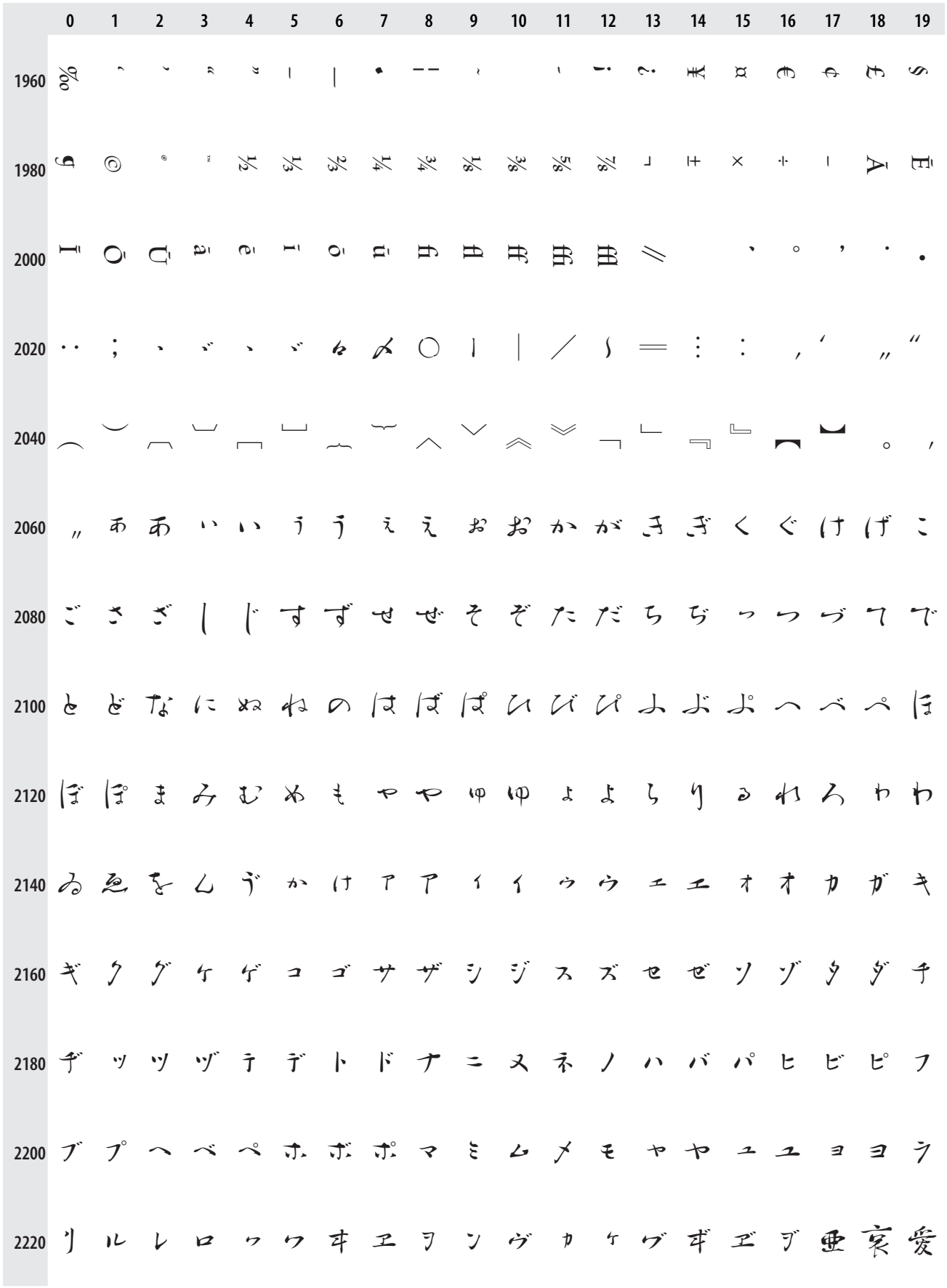

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
560
580
600
620
640
660
680
700
720
740
760
780
800
820

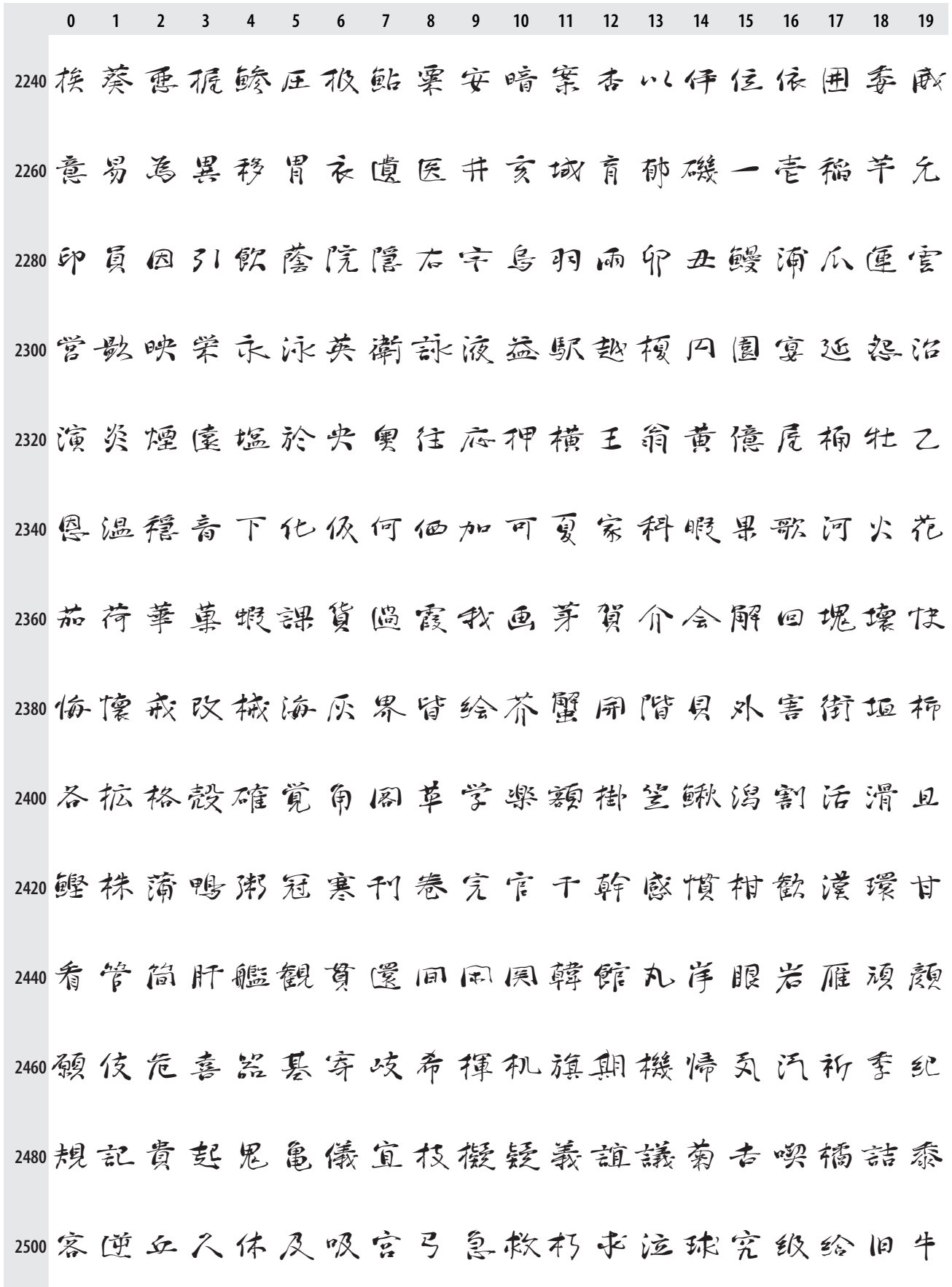

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
840
860
880
900
920
940
960
980
1000
1020
1040
1060
1080
1100

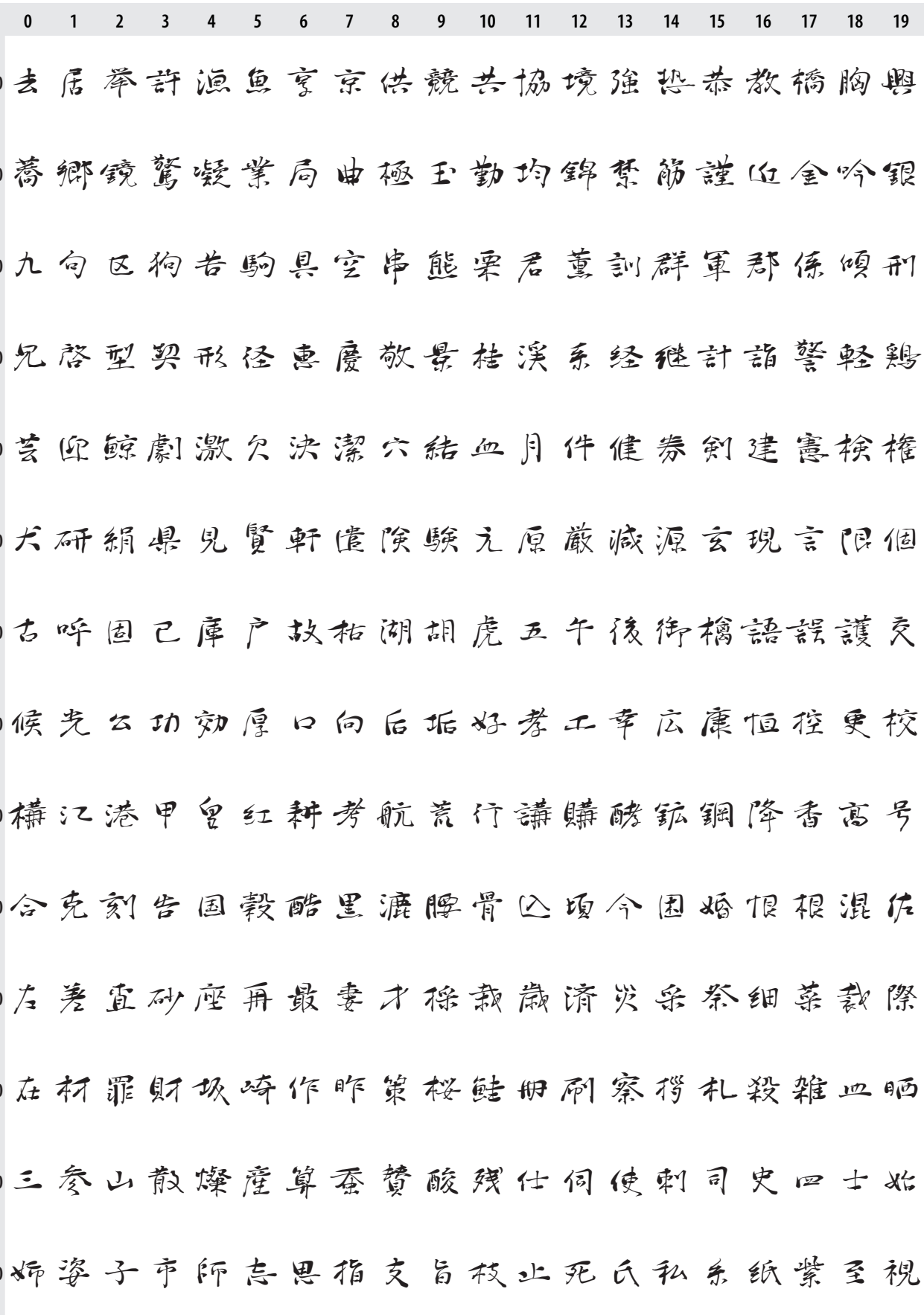

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1120
1140
1160
1180
1200
1220
1240
1260
1280
1300
1320
1340
1360
1380


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1400
1420
1440
1460
1480
1500
1520
1540
1560
1580
1600
1620
1640
1660

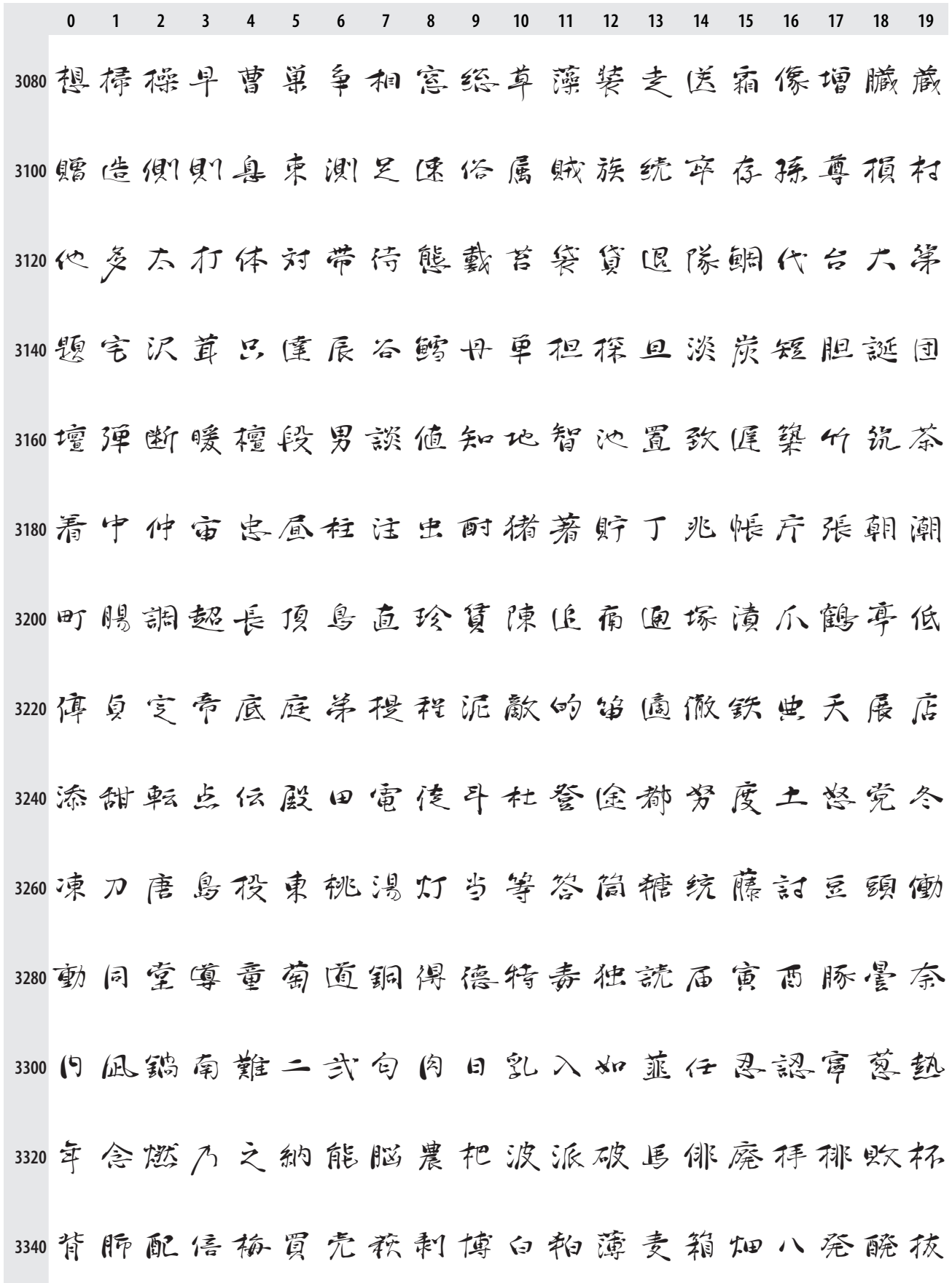

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1680
1700
1720
1740
1760
1780
1800
1820
1840
1860
1880
1900
1920
1940

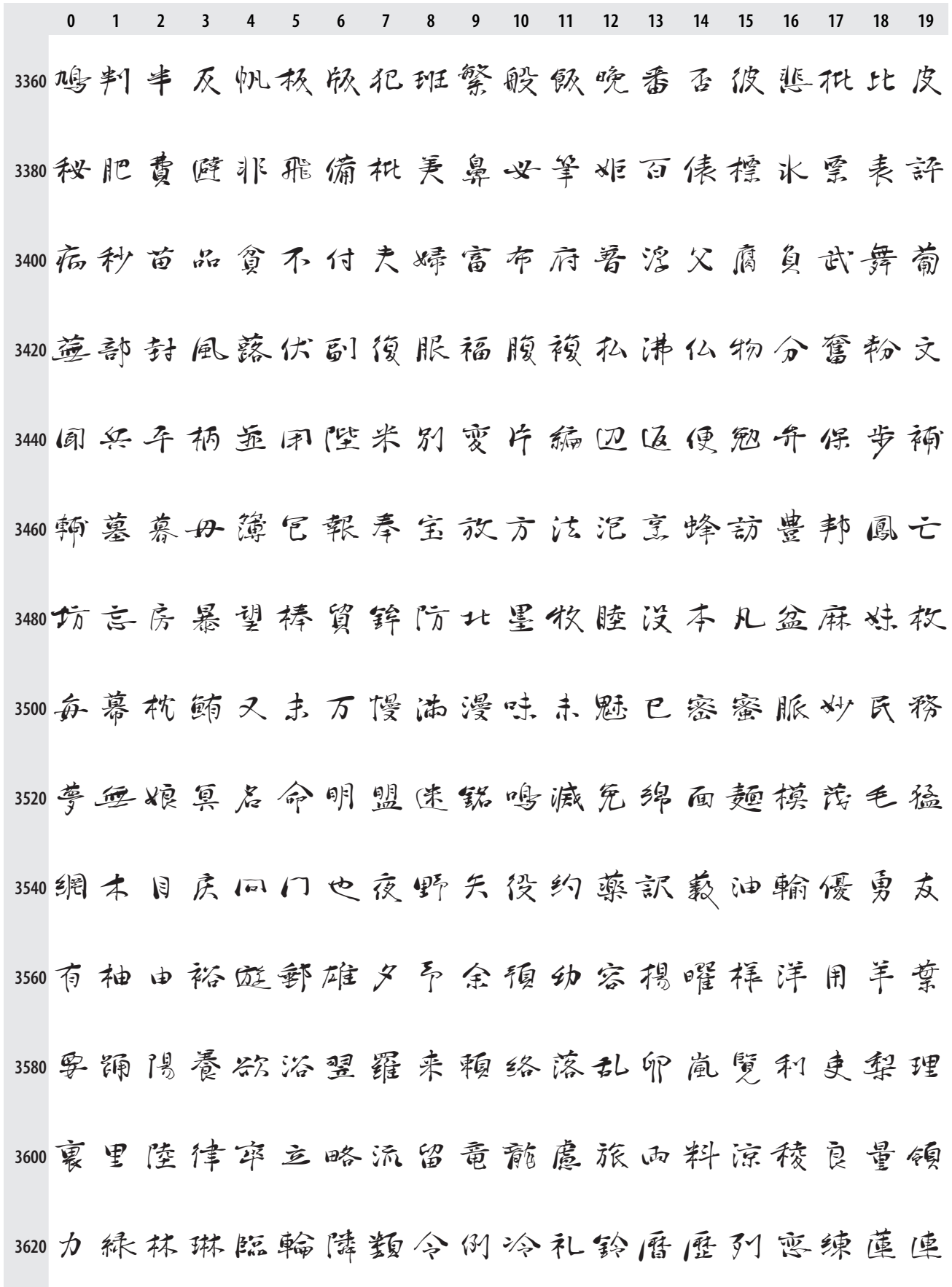

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1960
1980
2000
2020
2040
2060
2080
2100
2120
2140
2160
2180
2200
2220

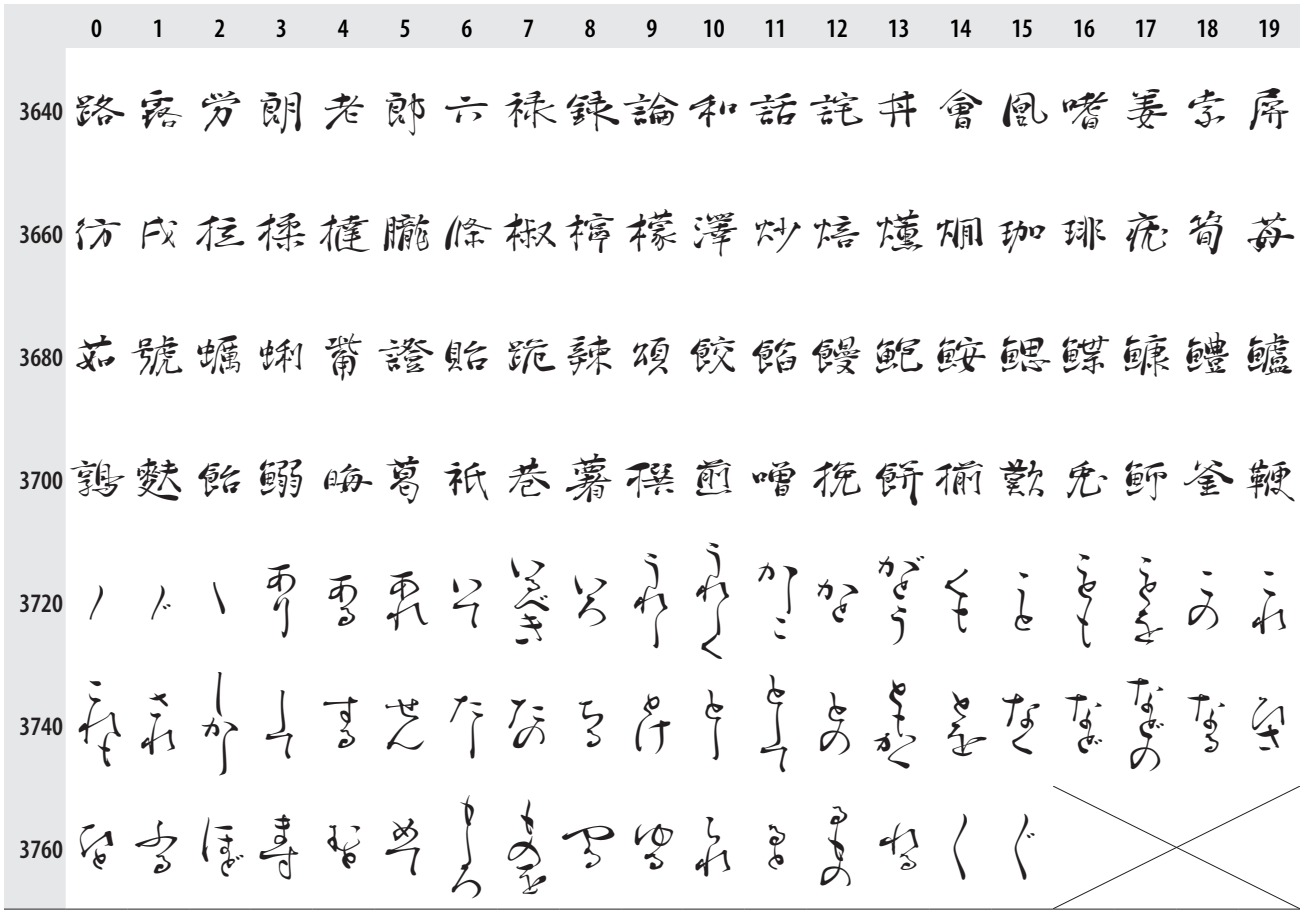

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2240
2260
2280
2300
2320
2340
2360
2380
2400
2420
2440
2460
2480
2500


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2520
2540
2560
2580
2600
2620
2640
2660
2680
2700
2720
2740
2760
2780


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2800
2820
2840
2860
2880
2900
2920
2940
2960
2980
3000
3020
3040
3060


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
3080
3100
3120
3140
3160
3180
3200
3220
3240
3260
3280
3300
3320
3340


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
3360
3380
3400
3420
3440
3460
3480
3500
3520
3540
3560
3580
3600
3620



Author

Dr. Ken Lunde, Senior Computer Scientist, CJKV Type Development, Adobe Systems Incorporated

Publishing Date

May 10, 2010

特殊用途の OpenType 日本語フォントチュートリアル : かづらき

1.1 はじめに

このチュートリアルは、特殊用途の OpenType® 日本語フォントの開発者を対象に、KazurakiSP2N-Light（以下、「かづらき®」と略称）を例にし、純粋なプロポーショナル日本語フォントの開発方法を説明するためのものです。本書で記述、参照、あるいはここに添付された技法、ツール、制御ファイルは、AFDKO (Adobe® Font Development Kit for OpenType) Version 2.0 またはそれ以降に含まれるツールと密接に関連しています。AFDKO は以下の URL から無料で入手可能です。

http://www.adobe.com/devnet/opentype/afdko/

本書の最後の項には、（本書発行時点において有効な）アプリケーションとの互換性に関する情報が掲載されていますので、特に注意してお読みください。

また、本書の内容についてご質問がありましたら、著者 Dr. Ken Lunde（ケン・ランディ）までご連絡ください。(lunde@adobe.com)

1.2 かづらきの制作意図

かづらきでは、縦組でしか用いない一部の平仮名の合字を除くすべての文字について、横組用と縦組用の両方の形状をフォントに持たせています。つまり、フォント中の各文字に対しそれぞれ 2 つのグリフを持たせています。グリフそのものは、2 つともまったく同じものですが、その字幅や、X 軸および Y 軸方向のデフォルトの位置設定に違いがあります。縦組用に、すべてのグリフを重複させて持たせる必要がありました。これは、OpenType の「vmtx」テーブル内でグリフを X 軸と Y 軸の双方に沿ってシフトする機能に制限があったことに加えて、アプリケーションが GSUB や GPOS の機能に依存せずに、これらの機能が有効でない場合でも、デフォルトで期待する動作が得られるようにしたいと考えたからです。

このフォントには、標準的な仮名文字がすべて含まれますが、漢字については、挨拶状やメニューの作成に用いられる、ごく限られた 1,483 文字だけしか含まれていません。さらに、かづらきには、キーボードによる入力を補助するプロポーショナルラテングリフの基本セットも含まれています。最小限のグリフ集合として、ASCII 文字 (U+0020 ～ U+007E) のグリフを含めることを推奨します。このように限定された利用範囲内では、かづらきは十分有効に機能し、またかづらきと同様のフォントを作成するための実例として役立ちます。

1.2.1 純粋なプロポーショナルグリフ

位置や向きが違うために日本語フォントでは縦組用と横組用とで異なるグリフが通常必要とされる、拗促音用の小仮名、一部の句読符号、括弧類の例外を除けば、かづらきに含まれるグリフの横組用と縦組用とは同一のものです。縦組用のグリフは、グリフデータを事後処理することによって生成され、Y 軸上の位置とメトリクスが異なります。

1.2.2 縦組用の平仮名合字

かづらきには縦組用の平仮名合字（51 文字）も少数含まれています。そのうちの 3 文字は 4 連の合字、12 文字は 3 連の合字、残りの 36 文字は 2 連の合字です。これらの合字は、GSUB 機能「liga」により起動されます。「liga」を用いる利点は、十分な文字組版機能を有するアプリケーションのほとんどで、その GSUB 機能が自動的に有効となるか、またはユーザが標準のユーザインターフェイスから有効にすることが可能になります。これにより、縦組用の平仮名合字が、多くのアプリケーションで特別な設定を行うことなく、使用可能になります。

1.2.3 CID キー方式と名前キー方式の構造

かづらきは、CID キー方式の OpenType フォントとして作成されました。かづらきの「CFF」テーブルは、CIDFont リソースから作成されたものです。かづらきは名前キー式のフォントとして作成することも可能でしたが、日本語フォントの場合、CID キー方式のフォントとした方が有利な点があります。第一の利点は、CID キー方式の構造が複数のヒント辞書に対応可能なことです。各ヒント辞書が個別のグリフクラスごとに有効に働き、各ヒント辞書はそれぞれ独自のヒント制御用のパラメータをもつことができます。複数のヒント辞書を使用できることが、レンダリング時に大きな利点となります。

1.3 OpenType テーブルの設定と上書き

かづらきの場合、多くの OpenType テーブルについて、特殊な設定をする必要があります。この項では、これらのテーブルの特殊な設定方法について個別に解説し、さらに、AFDKO のツール「makeotf」への入力ファイル「features」中で、テーブルの上書きをどのように指定するかについても解説します。

1.3.1 BASE

テーブル「BASE」においては、ICF（Ideographic Character Face: 漢字の字面）値とベースライン値を通常通り、標準に設定すること以外、特別な処理を行う必要はありません。ここで重要なのは、どの OpenType フォントでも「BASE」テーブルを必ず含める必要があるという点です。以下は、かづらきの「features」ファイルで「BASE」テーブルを上書きする例です。

```
table BASE {
  HorizAxis.BaseTagList              icfb icft ideo romn;
  HorizAxis.BaseScriptList DFLT ideo -117 877  -120 0,
                           hani ideo -117 877  -120 0,
                           kana ideo -117 877  -120 0,
                           latn ideo -117 877  -120 0;
  VertAxis.BaseTagList               icfb icft ideo romn;
  VertAxis.BaseScriptList  DFLT ideo 3    997  0    120,
                           hani ideo 3    997  0    120,
                           kana ideo 3    997  0    120,
                           latn ideo 3    997  0    120;
} BASE;
```

上記では、かづらきのグリフ集合に基づいて、「DFLT」、「hani」、「kana」、「latn」の文字の種別（script）だけが「BASE」テーブルの上書きを宣言している点に注意してください。

1.3.2 CFF

かづらきの本来のソースデータは、それぞれ 1,000 ユニットの字幅を持つ 1,827 のグリフを含む名前キー方式の OpenType フォントです。さらに、この同じソースフォントには、希望するグリフメトリクス（縦横の字幅の値と、X と Y 軸に沿ったシフト値）を指定する GPOS 機能「palt」と「vpal」も含まれます。また、これらの GPOS 機能は、縦組用グリフと横組用グリフのデフォルトのメトリクスのソースとしても使用されます。最終的に完成したフォントは、3,776 のグリフ（CID 収容範囲 0 〜 3775）を含む Adobe-Identity-0 CID キー方式の OpenType フォントとなっています。

初期のグリフの集合である 1,827 のグリフの処理には、AFDKO に搭載された 3 つのツール「tx」「mergeFonts」「rotateFont」を用います。

最初のツール「tx」は、ソースである OpenType フォントから「CFF」テーブルを抽出して、名前キー方式の Type 1 フォントに変換します。（以下のコマンドライン参照）

```
% tx -t1 KazurakiSource.otf > font.pfa
```

二つ目のツール「mergeFonts」は、グリフ名を CID 番号に変換し、同時に、元の横組用グリフから縦組用グリフを作成します。（以下のコマンドライン参照）

```
% mergeFonts -cid cidfontinfo cidfont.raw h.map font.pfa v.map font.pfa
```

この結果作成されたのが「cidfont.raw」という名前の Adobe-Identity-0 CIDFont リソースで、1 つのヒント辞書に 3,476 のグリフが収容されています。

三つ目のツール「rotateFont」は、横組用のグリフの字幅を設定し、X 軸に沿って各グリフをシフトします。これらの字幅と X 軸シフト値は、名前キー方式の OpenType ソースフォントの GPOS 機能「palt」内に収納されています。（以下のコマンドライン参照）

```
% rotateFont -t1 -rtf shift.map cidfont.raw cidfont-prop.raw
```

これによって出来上がったのが「cidfont-prop.raw」という名前の Adobe-Identity-0 CIDFont リソースで、これには、プロポーショナルメトリックスを持った横組用の 3,476 グリフが含まれています。さらに、300 のプロポーショルラテングリフがグリフ集合に追加され、合計 3,776 のグリフを収録しています。

平仮名の「し」を例として、横組用グリフの処理について詳しく見てみましょう。このグリフの元になる名前キー方式フォントのグリフ名は「CID864」（Adobe-Japan1-x 文字コレクションの CID+864 にちなんだもの）です。その名前キー方式の OpenType ソースフォント内における GPOS 機能「palt」の設定値は以下のようになっています。

```
position CID864 <-223 0 -485 0>;
```

ツール「mergeFonts」で処理した後、この（横組用）グリフは CID+224 になります。上記の「palt」データは、rotateFont が必要とする下記の指定値をマッピングファイル「shift.map」内に生成するために利用されます。

```
224 224 515 -223 0
```

この計算は簡単で、「-223」は、X 軸上のシフト値としてそのまま使われ、初期設定幅「1000」に「-485」を加算（実際には減算）した数値「515」が新しいデフォルトの字幅になります。

縦組用グリフの字幅と Y 軸シフト値は、「vmtx」テーブル中の定義を上書きして指定し、この上書きした定義を「features」ファイルに挿入します。縦組用グリフの字幅と Y 軸の位置決めに関連した処理については後述を参照ください。

これらのツールが実行され、各グリフに CID 番号が割り当てられ、横組用グリフの字幅と X 軸上の位置（プロポーショナル）のデフォルトの字幅値が設定された後は、通常通り CIDFont のヒンティングを行います。このヒンティングの過程で、複数のヒント辞書が生成されます（理想的には、各グリフクラスにつき一つのヒント辞書を生成）。

注： CIDFont 中に複数のヒント辞書を構築する工程は、AFDKO には含まれていないファイルやツールが必要となります。本書では、その方法についての解説は意図的に適宜省略されています。ただし、AFDKO の一環として配布される Adobe テクニカルノート #5900「AFDKO Version 2.0 チュートリアル : mergeFonts, rotateFont & autohint」中の「mergeFonts」に記載の方法によって、複数のヒント辞書を設定することが可能となります。mergeFonts を使って、複数のマッピングファイルのそれぞれ一行目でヒント辞書を指定するのが適切な方法です。実際、かづらきのために使用したアドビシステムズの独自ツールは、mergeFonts を利用して複数のヒント辞書を設定します。

特殊用途の Adobe-Identity-0 文字コレクション

かづらきのグリフ集合は Adobe-Japan1-6 文字コレクションに準拠していないうえ、この種の特殊用途のフォントの作成のために、Adobe-Japan1-6 文字コレクションを拡張することは意味がありません。そのため、CFF には、特殊用途の Adobe-Identity-0 文字コレクションであることが明記されています。「Adobe-Identity-0」では Kazuraki が日本語フォントとであることが明示されませんが、他のテーブルの設定やいくつかの点についてチェックしてみることによって、からすると、日本語フォントであることを確かめることができます。

基本的に、Adobe-Identity-0 文字コレクションを用いる利点は、あらかじめ決まった言語や文字体系を想定していないので、TrueType フォントや名前キー方式の OpenType フォントのように、ダイナミックなグリフセットに基づいた CIDFonts の構築が可能なことです。ただし、Adobe-Identity-0 文字コレクションを使う方法は、一般用途の OpenType 日本語フォントの生成には使用すべきではなく、そのような場合は、Adobe-Japan1-x 文字コレクションをご使用ください。

ファイルのサイズに関する注意点

横組 / 縦組のグリフの対は、アウトラインという点では同一であるため、AFDKO のツール「makeotf」のサブルーチン化機能を使うと、元のソースデータよりわずかに大きいだけの CFF テーブルを生成することができます。そのテーブルが含むグリフは、実際には約半分だけで済むこととなり、サブルーチン化した CFF テーブルのファイルは、サブルーチン化しない場合の約 50% の大きさにしかなりません。

ヒンティングに関する注意点

かづらきの場合、ステム幅という点ではヒンティングが通常通りに適用されますが、並び域においては通常とは異なります。仮名や漢字のような非ラテングリフクラスのヒント辞書には通常、以下のような /BlueValues 配列を用います。

```
/BlueValues [-250 -250 1100 1100] def
```

しかし、縦組専用の平仮名合字のバウンディングボックス（グリフの字面に内接する最少の正方形または長方形）が通常より大きい（背が高い）ため、グリフにこれらの仮名グリフのヒント辞書には異なる値を用いることが必要となります。かづらきの「Kana」ヒント辞書では以下の /BlueValues 配列を使用しています。

```
/BlueValues [-1250 -1250 2000 2000] def
```

さらに、かづらきの「Dingbats」と「Kanji」ヒント辞書は、その対象となるグリフの形状が 1000×1000 の全角正方形をの上下にいくぶんはみ出ているため、同一の /BlueValues 配列を使用しています。

以下は、かづらきの /FontBBox の内容を示したものです。

```
/FontBBox {-338 -1179 1587 1939} def
```

このように、かづらきの場合、少なくとも「Kana」ヒント辞書については、/BlueValues 値を –1179（Y 軸での最低位置）よりも低く、1939（Y 軸での最上の位置）よりも高くする指定する必要がありました。

1.3.3 GPOS

かづらきに含まれる GPOS 機能は、横組のカーニングを行う「kern」と、縦組のカーニングを行う「vkrn」だけです。元のソースデータに含まれる GPOS 機能「palt」と「vpal」は、横組 / 縦組の字幅と、X 軸方向での文字の位置のシフト量および Y 方向での文字の位置のシフト量を指定する工程のために用意されているものですが、こうした GPOS 機能はプロポーショナル書体のかづらきの場合には不要なため、生成後のフォントには含まれません。かわりに、これらの値はグリフのデフォルトのメトリクスを定義するために使用されます。

1.3.4 GSUB

かづらきには「fwid」、「vert」、「vrt2」及び「liga」という GPOS 機能が 4 つだけ含まれています。これらの機能の順序は普通、「liga」の後に「vert」と「vrt2」がきますが、このフォントの縦書き専用の平仮名合字については、この順序を逆にして、「vert」と「vrt2」の後に「liga」がくるようにする必要があります。一般に、「features」ファイル中の GPOS 機能と GSUB 機能の順序は重要です。AFDKO のツール「makeotf」が生成する「GPOS」と「GSUB」テーブルの順序にそれが反映されるからです。

GSUB 機能「vert」と「vrt2」は、横組用のグリフの形を縦組用のものに置き換えます。機能はフォントの大部分に適用されます。GSUB 機能「vert」と「vrt2」が実行されると、GSUB 機能「liga」を経由して、縦組専用の平仮名合字が適用されます。

縦組に対応する OpenType 準拠のアプリケーションは、自動的に GSUB 機能「vert」（存在する場合は「vrt2」）を実行します。これらのアプリケーションはまた、GSUB 機能「liga」をデフォルトで実行し、縦組専用の平仮名合字を起動します（あるいは、デフォルトに設定します）。

1.3.5 OS/2

かづらきには特殊用途の Adobe-Identity-0 文字コレクションが使われているため、OS/2.unicodeRange や OS/2.codePageRange フィールドなど、「OS/2」テーブルの一部のフィールドを注意深く設定する必要があります。かづらきの場合、「OS/2」テーブルを書き換えた以下の例に示されるような設定値を「features」ファイルに指定します。

```
XHeight 423;
CapHeight 645;
UnicodeRange 0 1 2 5 31 33 35 36 38 48 49 50 59 62 65 68;
CodePageRange 1252 932;
```

ここでは、「XHeight」と「CapHeight」の値が、グリフ集合に含まれるプロポーショナルラテングリフに対応した値に指定されていることに注意してください。

以下は、「UnicodeRange」値の説明です。

0	Basic Latin
1	Latin-1 Supplement
2	Latin Extended-A
5	Spacing Modifier Letters
31	General Punctuation
33	Currency Symbols
35	Letterlike Symbols
36	Number Forms
38	Mathematical Operators
48	CJK Symbols And Punctuation
49	Hiragana
50	Katakana
59	CJK Unified Ideographs
62	Alphabetic Presentation Forms
65	Vertical Forms
68	Halfwidth And Fullwidth Forms

「CodePageRange」の値 1252 は「Latin 1」、932 は「JIS/Japan」を示しています。

これらの「OS/2」テーブルの設定から、かづらきが明らかに日本語フォントであることが分かります。

1.3.6 VORG

テーブル「VORG」は、AFDKO のツール「makeotf」を使用すると自動的に生成され、テーブル「vmtx」から設定値とそれを上書きした値が抽出されます。テーブル「vmtx」の設定値や上書きした値に関する情報は、「vmtx」テーブルの項をご覧ください。

1.3.7 cmap

CID キー方式の OpenType フォントのテーブル「cmap」は、単一または複数の CMap リソースから作成されます。かづらきは特殊用途の Adobe-Identity-0 文字コレクションに基づいているため、特殊用途の CMap リソースが必要となります。縦組用グリフには GSUB 機能「vert」と「vert」を使ってアクセスできるため、横組用グリフだけが Unicode のコードポイントからマッピングされます。

1.3.8 name

テーブル「name」は、英語と日本語の文字列を適宜設定して通常通り作成します。ただし唯一の例外は「name.ID=20」の文字列で、かづらきが特殊用途であることから必要ではありません。テーブル「name」の文字列は、「FontMenuNameDB」と「features」の両ファイル中で指定されます。その際、必要に応じ、できるだけ多くの文字列に対して、文字の種別と言語が日本語であることを明示的に指定するよう心がけてください。

OpenType 日本語フォントのテーブル「name」のストリング設定に関する詳細は、Adobe テクニカルノート #5149（「OpenType-CID/CFF CJK Fonts: 'name' Table Tutorial」）をご参照ください。この文書は、以下の URL から無料で入手できます。

http://www.adobe.com/devnet/font/pdfs/5149.OTFname_Tutorial.pdf

1.3.9 vmtx

かづらきのようなフォントを作成する場合に、テーブル「vmtx」はきわめて重要な役割を果たします。縦組の字幅と Y 軸上のシフト値がこのテーブルに指定されるからです。テーブル「vmtx」に指定されている記述はすべてデフォルトの値となります。これにより、縦組に対応する OpenType 準拠のアプリケーションであれば修正をする必要なく、この種のフォントを使用できるようになります。

かづらきの「features」ファイルの中の上書きした「vmtx」テーブルのには、「VertAdvanceY」と「VertOriginY」のステートメントが数多く含まれています。これは縦組み用のグリフのほぼすべてで、それらのうちの一方または両方の「vmtx」の上書きが必要となったためです。

平仮名の「し」を例にして、縦組用グリフの処理について以下で、確かめてみることにします。このグリフの元になる名前キー方式のフォントのグリフ名は「CID864」（Adobe-Japan1-x 文字コレクションの CID=864 にちなんだもの）です。その GPOS 機能「vpal」の設定値は以下のようになっています。

```
position \CID864 <0 -26 0 331>;
```

ツール「mergeFonts」で処理した後、この（縦組）グリフは CID+2083 になります。上記の「vpal」データは、以下に示される「vmtx」テーブルを「features」ファイル内に生成するのに利用されます。

```
VertOriginY \2083 906;
VertAdvanceY \2083 1331;
```

これらの数値は簡単に算出できます。880（全角の正方形の上限を表す固定値）から「-26」を差し引くと、906 になり、これが新座標となります。さらに、デフォルトの字幅の 1000 ユニットに「331」を加えると、1331 になります。

1.4 特殊ツール

すべての制御ファイルとデータを一度の操作で生成できるように、Perl 言語で特殊用途のツールが書きました。「mergeFonts」や「rotateFont」を実行するときに使用するマッピングファイルはこのツールで作成されます。また、「vmtx」テーブルの上書きを含んだ「features」ファイルや、GSUB と GPOS のすべての機能一のあらゆる定義も同様にそのツールで作成しました。さらに、Unicode (UTF-32) の CMap リソースの作成に利用した元のデータもそのツールによるものです。グリフ数が多いうえ、その相互関係が複雑なため、手作業で行うのは煩雑であり、エラーを引き起こす原因となります。そのため、ツールの開発が必要でした。

この種のツールを開発する場合には、ソースとなるフォントのグリフ名から最終的な CID 番号へのマッピングを常に保持しておくと非常に便利なことが分かりました。こうすることで、CMap リソースの元のデータの生成がとても簡単になり、他の作業も容易に行えました。

1.5 OpenType 制御ファイルとデータ

名前から CID キーへの変換が完了したあとは、AFDKO のツール「makeotf」に必要となる通常の制御ファイルやデータの生成または準備を行う必要があります。これらの制御ファイルやデータについては次の項で説明します。

1.5.1 CIDFont リソース

かづらきの CIDFont リソースは、適切な数のヒント辞書（理想的には、各グリフクラスにつきそれぞれ一つのヒント辞書）と、Adobe-Identity-0 という ROS（/CIDSystemInfo 辞書の 3 項目である /Registry、/Ordering、/Supplement の略）を用いて、通常通りに構築されます。前述のように、かづらきの CIDFont リソースには、3,776 のグリフ（CID 収容範囲 0 ～ 3775）が含まれています。さらに、かづらきには、以下の 6 つのヒント辞書が搭載されています。

- KazurakiSP2N-Light-Dingbats (グリフ数 102)
- KazurakiSP2N-Light-Generic (グリフ数 1)
- KazurakiSP2N-Light-Kana (グリフ数 407)
- KazurakiSP2N-Light-Kanji (グリフ数 2,966)
- KazurakiSP2N-Light-Proportional (グリフ数 150)
- KazurakiSP2N-Light-ProportionalRot (グリフ数 150)

1.5.2「features」ファイル

「features」ファイルは、特定のテーブルを上書きすることができ、GPOS 機能と GSUB 機能を定義することができるなど、重要な役割を果たします。かづらきには 2 つの GPOS 機能「kern」と「vkrn」があり、それぞれ横組と縦組のカーニングをペアで指定しています。また、GSUB 機能としては「vert」と「liga」の 2 つが含まれており、前述のように、どちらの機能を先にするかが重要となります。最後に、テーブル「VORG」の作成にも使われる「vmtx」テーブルを上書きしたものが、縦組用のメトリックスのデフォルト値を指定します。

1.5.3「FontMenuNameDB」ファイル

OpenType フォントのテーブル「name」に記録されている英語と日本語のメニュー名は、「FontMenuNameDB」ファイルに記述します。かづらきの「FontMenuNameDB」項目は以下に示す通りです。

```
[KazurakiSP2N-Light]
   f=3,1,0x411,\304b\3065\3089\304d SP2N
   s=3,1,0x411,L
   l=3,1,0x411,\304b\3065\3089\304d SP2N L
   f=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N
   s=1,1,11,L
   l=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N L
   f=Kazuraki SP2N
   s=L
   l=Kazuraki SP2N L
```

ここで重要なことは、フォントを使用するアプリケーションが日本語のメニュー名を適切に認識できない場合を考慮して、日本語のメニュー名だけでなく、英語のメニュー名も必ず設定する必要があることです。

1.5.4 CMap リソース

かづらきのような特殊用途のフォントには、Unicode CMap リソースだけが必要となります。BMP 範囲外のマッピングがない場合でも、AFDKO のツール「makeotf」への入力ファイルとしては、UTF-32 CMap リソースの利用を推奨します。かづらき用の Unicode CMap リソースは、「UniKazurakiSP2N-UTF32-H」という名前のもので、かづらきフォント専用の組み合わせになっています。この CMap リソースは、AFDKO のツール「makeotf」への入力としてだけ利用され、最終的な OpenType フォントの Unicode「cmap」サブテーブルが構築されます。

CMap リソースの作成に関する詳細は、Adobe テクニカルノート #5099（「Building CMap Files for CID-Keyed Fonts」）をご参照ください。この文書は、以下の URL から無料で入手できます。

http://www.adobe.com/devnet/font/pdfs/5099.CMapFiles.pdf

1.6 検証と互換性に関する配慮

かづらきは、Adobe InDesign® CS2 上でも利用可能ですが、制約があります。所期の動作を行うことができます。横組用および縦組用のメトリクスを正しく認識し、縦組専用平仮名合字も含め、適切な縦組のレイアウトに対応します。

Adobe Illustrator® CS2 と Adobe Photoshop® CS2 でも、それは主に、GSUB 機能「liga」がオンになっていても、縦書き専用平仮名合字が正しく機能しないことにあります。

さらに、上述のアプリケーションや他のアプリケーションで、かづらきの日本語名がフォントメニューに表示されない場合があります。その場合テーブル「name」で指定された英語のメニュー名の文字列が、かづらきのフォント名の英語での表示に用いられることがあります。

かづらきは、InDesign CS3、Illustrator CS3、Photoshop CS3 など、CS3 のアプリケーションでは適切に機能します。かづらきや類似のフォントには、CS3 か CS4 アプリケーションをご使用になることを推奨します。

かづらきは、特殊（かつ限定的）なグリフ集合であるため、上記のアプリケーション等での合成フォント（Composite Font）機能の構成要素として使用することは推奨しません。

特殊用途の OpenType 日本語フォントを開発する際は、種々の OS 上で、様々なアプリケーションを用い、文書作成の全工程におけるフォントの厳密なテストを慎重に繰り返すことが肝要です。

1.7 グリフ概要

以降の 14 ページにおいて、かづらきの全ての 3,776 グリフを CID 番号順に並べています。次のリストでは CID レンジごとの内容を列挙します。

1 ～ 150	横組み用のグリフ — プロポーショナルラテン文字
151 ～ 1863	横組み用のグリフ — 日本語文字
1864 ～ 2013	CID 1 ～ 150 の回転されたグリフ
2014 ～ 3722	CID 151 ～ 1863 の縦組み用のグリフ
3723 ～ 3775	縦組用の平仮名合字とくの字点

特殊用途日文 OpenType 字库教程: Kazuraki

1.1 简介

本教程用以指导日文字库开发者构建特殊用途的日文 OpenType® 字库，以 KazurakiSP2N-Light（以下简称为 Kazuraki®）为例介绍如何制作真正完全变宽的日文字库。本文所述、引用或附带的技术、工具和控制文件与 AFDKO（Adobe® Font Development Kit for OpenType）2.0 版或更高版本中包含的工具紧密相关，AFDKO 可以在以下的 URL 免费获得：

http://www.adobe.com/devnet/opentype/afdko/

请特别留意本文最后部分的内容，该部分包含与应用程序兼容性有关的信息。

如果您对本文内容有任何疑问，请随时与作者 Ken Lunde (lunde@adobe.com) 联系。

1.2 Kazuraki 设计意图

除了竖排片假名连笔字，Kazuraki 中的其他字形都有对应的横排和竖排两种形式。也就是说，字库中的每个字符有两种字形。字形本身是相同的，但是其字幅和在 X 轴、Y 轴上的默认位置并不相同。由于在 OpenType‘vmtx’表中无法同时设置沿 X 轴、Y 轴方向的字形位移，再加上强烈希望将期望的操作设为默认状态，而不依赖于应用程序对 GSUB 或 GPOS 性能的支持，因此所有字形均需复制后才能用于竖排。

该字库包括全部的标准假名和一定数量的汉字（确切说是 1483 个），适合制作日文贺卡、菜单和用于其它特殊用途。Kazuraki 也包括一组基本的变宽拉丁字形以辅助键盘输入，我们建议至少要包含对应于 ASCII (U+0020 – U+007E) 的字形。Kazuraki 在这样的有限条件下不仅拥有全面的功能，还可以作为制作同类字库的范例。

1.2.1 真正的变宽字形

按照日文字库的常规要求，Kazuraki 中用于拗促音的小假名、一部分的标点符号和括弧符根据位置或方向不同采用了不同的横、竖排字形。除此之外，所有横、竖排字形对中的字形是完全相同的。字形数据经过后处理得到竖排字形，这些竖排字形各自位于 Y 轴的不同位置，并且有不同的量度。

1.2.2 竖排平假名连笔字

Kazuraki 包括少量竖排平假名连笔字（确切地说是 51 个），其中三个是四字符连笔字，十二个是三字符连笔字，剩下的三十六个是二字符连笔字，这些连笔效果通过使用 GSUB 的‘liga’特性来实现。使用‘liga’的优势在于多数专业排版应用程序会自动调用此 GSUB 功能，或者至少允许用户通过标准 UI(用户界面) 激活该功能。由此可见，竖排平假名连笔字可以在许多应用程序中默认使用。

1.2.3 CID-keyed 与 name-keyed 结构的比较

Kazuraki 是一款 CID-keyed OpenType 字库，其‘CFF’表是从 CIDFont 资源中构建而来的。虽然 Kazuraki 也可被构建为 name-keyed 字库，但对于日文字库而言 CID-keyed 字库更有优势。其主要优点是 CID-keyed 结构支持多个提示字典 (hint dictionary)，每个提示字典可以理想地涵盖特定的一类字形，并且在每个提示字典中可以设置不同的提示参数。由此可见，使用多个提示字典能体现极大的渲染优势。

1.3 OpenType 表的设置与覆盖

Kazuraki 中的许多 OpenType 表需要特殊设置。本节将逐步说明这些设置，同时也介绍如何在“features”文件中进行表覆盖（“features”文件是 AFDKO 的 makeotf 工具的输入文件)。

1.3.1 BASE

除了设置常规的 ICF（Ideographic Character Face：表意字框）和特定的基线值之外，‘BASE’表不需要其他特殊处理。在 OpenType 字库中包含‘BASE’表是非常重要的。以下是 Kazuraki 的“features”文件中的‘BASE’表覆盖：

```
table BASE {
  HorizAxis.BaseTagList               icfb icft ideo romn;
  HorizAxis.BaseScriptList DFLT ideo -117 877  -120 0,
                           hani ideo -117 877  -120 0,
                           kana ideo -117 877  -120 0,
                           latn ideo -117 877  -120 0;
  VertAxis.BaseTagList                icfb icft ideo romn;
  VertAxis.BaseScriptList  DFLT ideo 3    997  0    120,
                           hani ideo 3    997  0    120,
                           kana ideo 3    997  0    120,
                           latn ideo 3    997  0    120;
} BASE;
```

请注意，根据 Kazuraki 所涵盖的全部字形，在‘BASE’表覆盖中仅声明了‘DFLT’、‘hani’、‘kana’和‘latn’文字类型。

1.3.2 CFF

Kazuraki 的最初源数据是 name-keyed OpenType 字库，包含 1827 个 1000 单位宽度的字形。该源字库还包含 GPOS 的‘palt’和‘vpal’特性，用于指定特定的字形度量（分别为横、纵向字幅，X 轴和 Y 轴偏移值)，这些 GPOS 特性作为生成最终字库的横竖排字形默认度量的来源，最终字库是基于 Adobe-Identity-0 字符集的 CID-keyed Opentype 字库，包含了 3776 个字形（CID 0–3775)。

在处理原始 1827 个字形时使用了三个 AFDKO 工具：tx、mergeFonts 和 rotateFont。

第一个工具 tx 只是从源 OpenType 字库中提取‘CFF’表，并将其转换为 name-keyed Type 1 字库，命令行如下：

```
% tx -t1 KazurakiSource.otf > font.pfa
```

第二个工具 mergeFonts 将字形从 name-keyed 转换为 CID-keyed，同时由原横排字形合成竖排字形，命令行如下：

```
% mergeFonts -cid cidfontinfo cidfont.raw h.map font.pfa v.map font.pfa
```

最终输出的是一个基于 Adobe-Identity-0 字符集的 CIDFont 资源“cidfont.raw”，所包含的 3476 个字形仅使用了一个提示字典。

第三个工具 rotateFont 用于设置横排字形宽度和 X 轴偏移量。字宽和 X 轴偏移量从源 name-keyed OpenType 字库的‘palt’GPOS 特性中读取。命令行如下：

```
% rotateFont -t1 -rtf shift.map cidfont.raw cidfont-prop.raw
```

最终输出的是包含了 3476 个字形，名为“cidfont-prop.raw”的 Adobe-Identity-0 CIDFont 资源，其横排字形已被设置了变宽度量。另外增加了 300 个变宽拉丁字形，字形数目被扩展到了 3776 个。

让我们用实例探讨一下如何处理平假名字符“shi”(し) 的横排字形。源 name-keyed 字库中的字形名为“CID864”（以 Adobe-Japan1-x 的字符集 CID+864 命名），其‘palt’GPOS 特性在源 name-keyed OpenType 字库中设置如下：

```
position \CID864 <-223 0 -485 0>;
```

在经过 mergeFonts 工具处理后，此字形（横排字形）的 CID 值变成了 CID+224。上述‘palt’数据被用于生成 rotateFont 指令所需的“shift.map”映射文件：

```
224 224 515 -223 0
```

计算很简单，值“-223”被用作 X 轴位移值，默认字幅值 1000 加上值“-485”后（减法运算）变成 515。

在“features”文件中添加“vmtx”表覆盖,在该表覆盖中设定字幅宽度和竖排字形的 Y 轴偏移值。有关竖排字形的字幅和 Y 轴位置的控制将在本文后面说明。

一旦这些工具开始运行，每个字形都被分配了相应的 CID，横排字形被设置了默认的字宽和 X 轴位置（也就是变宽字形），按惯例紧接着需要对 CIDFont 增加提示信息控制。增加提示信息控制的过程也涉及到创建多个提示字典，理想状态是每种字形类对应一个提示字典。

注意：在一个 CIDFont 中建立多个提示字典需要用到的工具不包含在 AFDKO 中，本文刻意省略了关于它们的描述。然而，在 AFDKO 附带的文档 Adobe Tech Note #5900（“AFDKO Version 2.0 教程：mergeFonts、rotateFont 和 autohint”）中提到的 mergeFonts 技术可用来建立多个提示字典。针对多个 mergeFonts 映射文件，建议在每个文件的首行命名一个提示字典，而且多个 mergeFonts 映射文件可以为单个提示字典指定字形。事实上，我们就是使用 mergeFonts 工具为 Kazuraki 建立了多个提示字典。

特殊用途的 Adobe-Identity-0 字符集

因为 Kazuraki 的整套字形不依附于 Adobe-Japan1-6 字符集，而且为该特殊用途的字库扩展 Adobe-Japan1-6 是没有多大意义的，所以在 CFF 中，将其登记为特殊用途的 Adobe-Identity-0 字符集。尽管“Adobe-Identity-0”没有明确指明 Kazuraki 是日文字库，但其它表的设置加上可靠的启发式方法可以表明它是日文字库。

本质上来说，使用 Adobe-Identity-0 字符集的优点是没有设置固定的语言或文字形式，这样就可以像 TrueType 和 name-keyed OpenType 字库一样，根据动态字形集构建 CIDFont。制作一般用途的日文 OpenType 字库不应使用 Adobe-Identity-0 字符集的技术，而应使用 Adobe-Japan1-x 字符集。

文件大小问题

因为一对横竖排字形在外形上完全相同，经过 AFDKO 的 makeotf 工具的子程序化之后，所生成的 CFF 表只比原始输入数据稍大，而原始输入数据仅包含了大约一半的字数，因此子程序化后的‘CFF’表的大小是未子程序化‘CFF’表的 50%。

Hinting 问题

对于 Kazuraki 而言，字体主干宽度的 Hinting（提示控制）设置与常规方法是相同的，但是对齐区域方面的设置是不同的。非拉丁字形类 (如假名和汉字) 的提示字典一般使用以下 /BlueValues 排列。

```
/BlueValues [-250 -250 1100 1100] def
```

但是竖排的平假名连笔字边界框比普通的字框要大（高），所以其提示字典需要不同的值，以确保对齐区域没有与其字形接触。

Kazuraki 的“Kana”提示字典使用以下 /BlueValues 排列：

```
/BlueValues [-1250 -1250 2000 2000] def
```

此外，Kazuraki 的“Dingbats”和“Kanji”提示字典也使用相同的 /BlueValues 排列，这是由于其字形形状的扩展范围在 1000×1000 字面大小上下。

以下为 Kazuraki 的 /FontBBox：

```
/FontBBox {-338 -1179 1587 1939} def
```

因此，要选择小于 –1179（Y 轴的最低位置）和大于 1939（X 轴的最高位置）的 /BlueValues 值，尤其对于“Kana”提示字典，这点至关重要。

1.3.3 GPOS

Kazuraki 包含的 GPOS 性能是‘kern’和‘vkrn’，分别用于横、纵向字距调整。源数据中的‘palt’和‘vpal’GPOS 性能用于驱动制作过程，以指定横排或竖排字幅和 X 轴、Y 轴方向的偏移值。由于‘palt’和‘vpal’GPOS 特性不是必需的，所以没有包含在最终字库中，这些值用于设定默认字形度量。

1.3.4 GSUB

Kazuraki 只包含四个 GSUB 特性：‘fwid’、‘vert’、‘vrt2’和‘liga’，这些特性的常规执行顺序是‘liga’排 在‘vert’和‘vrt2’之前，但是该字库的竖排平假名连笔字要求必须有不同的顺序，确切地说是‘liga’在‘vert’和‘vrt2’之后执行。一般来说，“features”文件中 GPOS 和 GSUB 性能的排序很重要，因为表中的排序将直接反映在 AFDKO 的 makeotf 生成的‘GPOS’和‘GSUB’表中。

‘vert’GSUB 性能可以用竖排字形替换横排字形，大部分字库都包含该功能。一旦使用了‘vert’GSUB 性能，竖排平假名连笔字效果就可以随后通过‘liga’GSUB 性能来实现。

支持竖排的 OpenType-savvy 应用程序（支持 OpenType 特性的应用程序）在竖排时将自动调用‘vert’（或‘vrt2’，如果有）GSUB 性能。这些相同的应用程序也将默认地调用‘liga’GSUB 功能，以激活（或设为默认）全竖排平假名连笔字。

1.3.5 OS/2

因为 Kazuraki 使用的是特殊用途的 Adobe-Identity-0 字符集，所以必须更加仔细地指定某些‘OS/2’表字段，如 OS/2.unicodeRange 和 OS/2.codePageRange 字段。在 Kazuraki 中，需要指定“features”文件中‘OS/2’表覆盖的以下字段：

```
XHeight 423;
CapHeight 645;
UnicodeRange 0 1 2 5 31 33 35 36 38 48 49 50 59 62 65 68;
CodePageRange 1252 932;
```

请注意，“XHeight”和“CapHeight”值被设置为其对应字形的变宽拉丁字形的值。

“UnicodeRange”值对应情况如下所示：

0	Basic Latin
1	Latin-1 Supplement
2	Latin Extended-A
5	Spacing Modifier Letters
31	General Punctuation
33	Currency Symbols
35	Letterlike Symbols
36	Number Forms
38	Mathematical Operators
48	CJK Symbols And Punctuation
49	Hiragana
50	Katakana
59	CJK Unified Ideographs
62	Alphabetic Presentation Forms
65	Vertical Forms
68	Halfwidth And Fullwidth Forms

“CodePageRange”中的值 1252 对应“Latin 1”，932 对应“JIS/Japan”。

这些‘OS/2’表中的参数设置有助于将 Kazuraki 明确确定为日文字库。

1.3.6 VORG

使用 AFDKO 的 makeotf 工具时，‘VORG’表将自动生成，而且是由“vmtx”表的设置和覆盖派生而来的。请参阅“vmtx”表这节来了解更多有关“vmtx”表设置和覆盖信息。

1.3.7 cmap

在生成 CID-keyed OpenType 字库的‘cmap’表时，需要用到一个或多个 CMap 资源。对于 Kazuraki 而言，由于它基于特殊用途的 Adobe-Identity-0 字符集，所以需要准备特殊用途的 CMap 资源。由于可以通过‘vert’和‘vrt2’GSUB 性能调用竖排字形，所以只需为横排字形映射 Unicode 编码。

1.3.8 name

按常规方式建立‘name’表，设置恰当的英文或日文字符串。唯一的例外是鉴于 Kazuraki 的特殊性，可以不设置 name.ID=20 的字符串。在 "FontMenuNameDB" 和“features”文件中详细设置‘name’表字符串时，请明确指明文字和语言是日文，尽可能恰当地设置众多的字符串。

关于日文 OpenType 字库‘name’表字符串的更多特殊设置信息，请参阅以下 URL 提供的 Adobe 技术说明 #5149（“OpenType-CID/CFF CJK Fonts: ‘name’ Table Tutorial”）：

http://www.adobe.com/devnet/font/pdfs/5149.OTFname_Tutorial.pdf

1.3.9 vmtx

“vmtx”表在构建字库（如 Kazuraki）时有着非常重要的作用，因为在该表中指定了竖排字形的字幅以及 Y 轴位移值，而这些值都将被看作默认操作值。因此支持竖排功能的 OpenType-savvy 应用程序可以无障碍地使用该类字库。

Kazuraki 的“features”文件在其‘vmtx’表覆盖中包含大量“VertAdvanceY”和“VertOriginY”设置语句，几乎每个竖排字形都至少需要设置其中一个的值。

让我们用实例探讨一下如何处理平假名字符“shi”（し）的竖排字形。源 name-Keyed 字库的字形称为“CID864”（以 Adobe-Japan1-x 字符集的 CID+864 命名）。其“vpal”GPOS 特性设置如下：

```
position \CID864 <0 -26 0 331>;
```

在经过 mergeFonts 工具处理后，此竖排字形变成 CID+2083。上述“vpal”数据用于重置“features”文件中的“vmtx”表，其设置如下所示：

```
VertOriginY \2083 906;
VertAdvanceY \2083 1331;
```

计算很简单：从 880（em-box 字体框上方的固定值）中减去值“-26”，得到的 906 即为新的原点值，值“331”加上默认的 1000 个单位的字幅，得到 1331。

1.4 特殊工具

用 Perl 编写的特殊用途专用工具可以一次生成所有控制文件和数据，控制 mergeFonts 和 rotateFont 工具执行的映射文件就是由此工具生成的。“features”文件所包含的‘vmtx’表覆盖及所有 GSUB 和 GPOS 特性的设置也是用该工具生成的。另外，构建 Unicode (UTF-32) CMap 资源的原始数据同样由此工具生成。由于字形数目多，且字形之间关系复杂，所以创建一个工具执行此操作非常重要，因为手动执行操作比较复杂而且容易出错。

在编写类似工具时，本人发现经常维护原始字库中的字形名称与最终 CID 的映射关系是非常有用的。这使得生成 CMap 资源原始数据更容易，也让其它任务变得更容易。

1.5 OpenType 控制文件和数据

完成 name-keyed 至 CID-keyed 的转换之后，需要生成或提供 AFDKO 的 makeotf 工具所需的常用控制文件及数据。这些控制文件和数据在下一节有详细说明。

1.5.1 CIDFont 资源

Kazuraki 的 CIDFont 资源按常规的方式建立，它包含适当数量的提示字典（理想状态是每类字形对应一个提示词典），以及登记在 ROS（/Registry、/Ordering 和 /Supplement，即 /CIDSystemInfo 字典的三个条目）中的 Adobe-Identity-0。如前文所述，Kazuraki 的 CIDFont 资源包含 3776 种字形 (CID 0-3775)。Kazuraki 确切包含 6 个提示字典，名称如下所示，字形数目在括号中：

- KazurakiSP2N-Light-Dingbats（102 个字形）
- KazurakiSP2N-Light-Generic（1 个字形）
- KazurakiSP2N-Light-Kana（407 个字形）
- KazurakiSP2N-Light-Kanji（2966 个种字形）
- KazurakiSP2N-Light-Proportional（150 个字形）
- KazurakiSP2N-Light-ProportionalRot（150 个字形）

1.5.2 “features” 文件

“features”文件有非常重要的作用，在其中可进行特定表的替换，并且可定义 GPOS 和 GSUB 特性。 Kazuraki 包含两个 GPOS 特性 ，即“kern”和“vkrn”，它们分别用来设定水平和垂直字距。还包括四个 GSUB 特性—“fwid”、“vert”、“vrt2”和“liga”,如前文所述其相对顺序很重要。最后是“vmtx”表覆盖，它也被用于构建“VORG”表，主要用于设置默认垂直度量。

1.5.3 “FontMenuNameDB” 文件

OpenType 字库的“name”表中的英文和日文菜单名称是在“FontMenuNameDB”文件中指定的。Kazuraki 的“FontMenuNameDB”条目如下所示：

```
[KazurakiSP2N-Light]
   f=3,1,0x411,\304b\3065\3089\304d SP2N
   s=3,1,0x411,L
   l=3,1,0x411,\304b\3065\3089\304d SP2N L
   f=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N
   s=1,1,11,L
   l=1,1,11,\82\a9\82\c3\82\e7\82\ab SP2N L
   f=Kazuraki SP2N
   s=L
   l=Kazuraki SP2N L
```

需强调的是，除了设置日文菜单名称，还必须设置英文菜单名称，以免在应用程序中使用这些字库时，其机制无法正确使用日文菜单名称。

1.5.4 CMap 资源

对于 Kazuraki 这样的特殊用途的字库，只需要 Unicode CMap 资源。即使在 BMP (Basic Multilingual Plane) 外没有映射，我们还是建议将 UTF-32 CMap 资源作为 AFDKO makeotf 工具的输入。对于 Kazuraki，为了与该字库紧密结合，Unicode CMap 资源被命名为“UniKazurakiSP2N-UTF32-H”。此 CMap 资源仅用作 AFDKO makeotf 工具的输入，以构建所生成 OpenType 字库的 Unicode“cmap”子表。

有关构建 CMap 资源的更多信息，请参阅以下 URL 提供的 Adobe 技术说明 #5099（“Building CMap Files for CID-Keyed Fonts”）：

http://www.adobe.com/devnet/font/pdfs/5099.CMapFiles.pdf

1.6 测试及兼容性考量

Kazuraki 可在 Adobe InDesign® CS2 及更高版本中正常工作，这些软件支持水平和垂直度量、恰当的竖排布局和竖排平假名连笔字功能。

Kazuraki 的功能在 Adobe Illustrator® CS2 和 Adobe Photoshop® CS2 中会有一定的限制，确切地说是即使启用了“liga”GSUB 功能，也不能切换出竖排平假名连笔字。

另外，这些应用程序和其它应用程序可能不会在其字体菜单中显示 Kazuraki 的日文菜单名称，代为显示的是“name”表中设定的英文菜单名称字符串。

Kazuraki 与 CS3 和 CS4 应用程序配合地非常好，特别是 InDesign、Illustrator 和 Photoshop。事实上，我们建议在 CS3 和更高版本的应用程序上使用 Kazuraki 及类似字库。

鉴于其独特（及有限的）的字形集，不建议将 Kazuraki 作为组件用于这些应用程序的复合字体功能中。

在开发特殊用途的日文 OpenType 字库时，慎重的做法是严格地在各种操作系统和应用程序中测试该字库，以覆盖整个文档编制工作流。

1.7 字形总览

以下 14 页按 CID 顺序排列了 Kazuraki 的全部 3776 个字形。以下列表提供了特定 CID 范围的信息：

1–150	横排字形—变宽拉丁文
151–1863	横排字形—日文
1864–2013	CID 1–150 预旋转字形
2014–3722	CID 151–1863 竖排字形
3723–3775	竖排平假名连笔字和双假名叠代符。