Infoprint Fonts

Creating User-defined Characters

G544-5854-00

Infoprint Fonts

IBM

Creating User-defined Characters

G544-5854-00

Note!

Before using this information and the product it supports, read the information in "Notices" on page 45.

First Edition (December 2002)

This edition of *IBM Infoprint Fonts: Creating User-defined Characters* applies to IBM Infoprint Fonts for Multiplatforms, Version 1 Release 1 Modification 0, program number 5648–E77; and to all subsequent releases of this product until otherwise indicated in new releases or technical newsletters.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there. Many of the IBM Printing Systems Division publications are available from the web page listed below.

Internet

Visit our home page at: `http://www.ibm.com/printers`

A Reader's Comments form is provided at the back of this publication. If the form has been removed, you can send comments by fax to 1-800-524-1519 (USA only) or 1-303-924-6873; by E-mail to `printpub@us.ibm.com`; or by mail to:

IBM Printing Systems Division
Department H7FE Building 004M
Information Development
PO Box 1900
Boulder CO 80301-9191 USA

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Chapter 1. Introduction

Type Transformer, a part of IBM® Infoprint® Fonts, is a program that transforms IBM Type 1 format and CID-keyed format outline fonts into fonts that can be used with IBM's advanced function printers. The following utilities are included with Type Transformer to enable you to add user-defined characters (UDCs) to the Japanese, Korean, Simplified Chinese, and Traditional Chinese fonts in IBM Infoprint Fonts:

- The FontLab graphic character editor
- The AFP™ Font Editor for code pages and coded fonts
- Sample code pages and batch files
- The AFP2FON utility, which converts raster fonts to outline fonts
- The CID2EPS utility, which extracts a a character from a font and presents it as an EPS image
- The DUVRMARK utility, which changes the date and time stamp or capture setting of a font

Notes:

1. Type Transformer was **formerly** part of AFP Font Collection.
2. Adobe Type 1 format outline fonts are hereafter referred to as *Type 1 fonts*. Adobe CID-keyed format outline fonts, also called Type 0 format fonts, are hereafter referred to as *CID-keyed fonts*.

This publication should be used by the font administrator when adding UDCs to double-byte character set (DBCS) fonts for IBM's advanced function printers. You should have a background in using Windows and an understanding of how fonts are used for printing on advanced function printers. You may need help from the system programmer.

After you install IBM Infoprint Fonts, you may need to set up your host system for Type Transformer.

Contents of this publication

This publication contains the following information:

- Chapter 1, "Introduction" briefly describes Type Transformer, explains how this publication is organized, and lists some related publications.
- Chapter 2, "CMaps, CIDFonts, and CID-keyed fonts" on page 3 describes the format and contents of CMaps and CIDFonts, the two parts of CID-keyed fonts, and explains how a CMap and CIDFont work together.
- Chapter 3, "Creating user-defined characters" on page 17 contains detailed instructions for adding a user-defined character to a CID-keyed font.
- Chapter 4, "Converting AFP raster UDCs to Type 1 outline UDCs" on page 41 explains how to convert a user-defined character from raster format to Type 1 outline format.

Related publications

For a list of related IBM publications and CD-ROMs, please see *IBM Infoprint Fonts: Font Summary*, G544-5846.

These Adobe Systems Incorporated publications describe Adobe fonts:

- *Adobe CMap and CIDFont Files Specification*, Technical Specification #5014, Version 1.0, (Mountain View, CA: Adobe Systems Incorporated, 16 October 1995)
- *Adobe Type 1 Font Format*, Version 1.1 (Reading, MA: Addison-Wesley Publishing Company, Inc., 1990), ISBN 0-201-57044-0
- *PostScript Language Reference Manual*, 2nd edition (Reading, MA: Addison-Wesley Publishing Company, Inc., 1990), ISBN 0-201-18127-4

FontLab User's Guide describes how to use FontLab to design a character.

Chapter 2. CMaps, CIDFonts, and CID-keyed fonts

Before you can create user-defined characters (UDCs), you must understand the CID-keyed fonts to which you will add your characters. This chapter:

- Describes the format and contents of CMaps and CIDFonts, the two parts of CID-keyed fonts
- Explains how a CMap and CIDFont work together

CMaps

A CMap is an ASCII text file that contains PostScript language instructions for mapping character codes to CIDs.

Format

A complete CMap contains the following sections:

- Header comments
- Initialization
- Identification
- Writing mode
- Codespace definitions
- Mapping information
- Closing code

Header comments

To ensure correct processing in all environments, every CMap file must begin with the comment characters %! (percent sign and exclamation point). Comment lines after the first begin with the characters %% (two percent signs).

A typical CMap comment section looks like this:

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: procset (CIDInit)
%%IncludeResource: procset (CIDInit)
%%BeginResource: CMap (CS-2093-H)
%%Title: (CS-2093-H IBM Japan2 0)
%%CreationDate: 2002-08-27 00:00:56
%%Version: 1
%%Copyright: (c) Copyright 2002 IBM Corporation, all rights reserved.
```

This comment:

```
%!PS-Adobe-3.0 Resource-CMap
```

identifies the file as a CMap that conforms to the conventions of PostScript Version 3.0.

These comments:

```
%%DocumentNeededResources: procset (CIDInit)
%%IncludeResource: procset (CIDInit)
```

identify a system support file, called CIDInit, to spoolers and parsers. The %%DocumentNeededResources comment indicates that CIDInit is a required external resource. The %%IncludeResource comment indicates that if CIDInit is not available in the PostScript interpreter, it should be included inline.

This comment:

```
%%BeginResource: CMap (CS-2093-H)
```

informs spoolers and parsers that this file is a CMap resource named CS-2093-H. There is a corresponding %%EndResource comment at the end of the file.

This comment:

```
%%Title: (CS-2093-H IBM Japan2 0)
```

states the CMap name (CS-2093-H), registry name (IBM), ordering name (Japan2), and supplement number (0).

This comment:

```
%%CreationDate: 2002-08-27 00:00:56
```

gives the date and time the CMap was created.

This comment:

```
%%Version: 1
```

states the version number. This is the same as the version number in the identification section of the CMap.

This comment:

```
%%Copyright: (c) Copyright 2002 IBM Corporation, all rights reserved.
```

is the copyright statement.

Initializing the CMap

The next three lines initialize the CMap:

```
/CIDInit /Procset findresource begin

12 dict begin

begincmap
```

The findresource begin operator directs the system to find the CIDInit file. There is a matching end operator near the end of the CMap file.

The begin operator defines a PostScript language dictionary. This one contains 12 elements in order to allow for VM processing.

The begincmap operator defines the CMap. There is a matching endcmap operator near the end of the CMap file.

Identifying the CMap

The next section identifies the character collection on which the CMap is based, the name of the CMap, and the version and type of the CMap:

```
/CIDSystemInfo 3 dict dup begin
   /Registry (IBM)      def
   /Ordering (Japan2) def
   /Supplement 0       def
end def

/CMapName /CS-2093-H def
/CMapVersion 1 def
/CMapType 0 def

/XUID [116 10 14204] def
```

The first group of def operators identifies the character collection: JapanIBM, created by IBM, initial ordering. The character collection must have the same registry and ordering as the character collection on which the CIDFont is based.

The CMapName line gives the name of the CMap. This is the same name as in the header comments.

The CMapVersion and CMapType comments identify the version and type of the CMap. The version is the same version as in the header comments.

CMapName and CMapType are required. CMapVersion is optional.

The XUID is an array of numbers that together form a unique identifier for the CMap. The first number identifies the organization that created the CMap: 116 is IBM. XUID is optional.

Writing direction

The next section indicates whether the CID-keyed font is written horizontally from left to right (0) or vertically from top to bottom (1):

```
/WMode 0 def
```

The WMode value in the CMap overrides the WMode value in the CIDFont.

Codespace definitions

The next group of sections defines the range of valid input character codes. This example shows a single codespace definition section:

```
3 begincodespacerange
  <4040> <68FE>
  <B341> <DAFE>
  <DC41> <ECFE>
endcodespacerange
```

The first line indicates that this section defines three valid ranges of character codes. The next three lines define the ranges by giving the beginning and end of each range as hexadecimal values.

You can define single-byte and double-byte character codes in the same CMap; but IBM recommends creating two separate CMaps.

Single-byte codespace ranges are *linear*; any value from the beginning to the end of the range, inclusive, is valid.

Double-byte codespace ranges are *rectangular*. It is easiest to visualize what this means by writing out a range of double-byte values in a block, as in Figure 1. The beginning and end of the range define the corners of a highlighted rectangle of valid codes.

```
4000...4020...4040 4041...4060...4080...40A0...40C0...40E0...40FE 40FF
4100...4120...4140 4141...4160...4180...41A0...41C0...41E0...41FE 41FF
4200...4220...4240 4241...4260...4280...42A0...42C0...42E0...42FE 42FF
4300...4320...4340 4341...4360...4380...43A0...43C0...43E0...43FE 43FF
   ⋮
6600...6620...6640 6641...6660...6680...66A0...66C0...66E0...66FE 66FF
6700...6720...6740 6741...6760...6780...67A0...67C0...67E0...67FE 67FF
6800...6820...6840 6841...6860...6880...68A0...68C0...68E0...68FE 68FF
6900...6920...6940 6941...6960...6980...6969...69C0...69E0...69FE 69FF
```

Figure 1. A rectangular codespace range: 4141 to 68FE

In other words, a value is not necessarily valid because it is greater than the beginning value and less than the end value. The first byte must fall between the first bytes of the beginning and end values; and the second byte must fall between the second bytes of the beginning and end values. 67A0 is valid because 67 is between 41 and 68 and A0 is between 41 and FE. 6820 is not valid because, although 68 is between 41 and 68, 20 is not between 41 and FE.

Each codespace definition section ends with an endcodespacerange operator.

A single codespace definition section can define up to 100 codespace ranges. If you need more than 100 definitions, code multiple definition sections.

Codespace ranges cannot overlap.

Mapping information

The next group of sections is the heart of the CMap: the character mappings. This example shows the beginning and end of a single mapping section:

```
100 BEGINCIDRANGE
<4040> <4040> 633
<4141> <4158> 1035
<4161> <4178> 1011
<4180> <41A0> 1092
<41B1> <41BA> 8092
   ⋮
<43A1> <43A1> 665
<43A2> <43A2> 980
<43A3> <43A3> 983
ENDCIDRANGE
```

The first line indicates that this section defines 100 mapping ranges. The next 100 lines, of which only the first 5 and the last 3 are shown, each define a mapping range by defining the beginning and end of a range of character codes and the beginning of the range of CIDs that are assigned in sequence to those codes. For example, this line:

```
<41B1> <41BA> 8092
```

translates into this mapping:

Table 1. Mapping character codes to CIDs

Character code	CID
41B1	8092
41B2	8093
41B3	8094
41B4	8095
41B5	8096
41B6	8097
41B7	8098
41B8	8099
41B9	8100
41BA	8101

Each mapping definition section ends with an endcidrange operator.

A single mapping definition section can define up to 100 mapping ranges. If you need more than 100 definitions, code multiple definition sections.

Mapping ranges, unlike codespace ranges, can overlap. The last mapping in the CMap overrides all preceding mappings for that character code.

Every character code mapped must fall within a valid codespace.

All the character codes in a range are mapped to the same character, 1, not a range beginning with 1.

Closing the CMap

The final section in the CMap establishes the CMap resource and closes the CMap file:

```
endcmap
CMapName currentdict /CMap defineresource pop
end
end
%%EndResource
%%EOF
```

The endcmap operator corresponds to the begincmap operator in the initialization section.

This line:

```
CMapName currentdict /CMap defineresource pop
```

explicitly states the encoding for the CMap file, defines the file as a CMap resource, and pops it off the stack.

The two end operators correspond to the dict begin operator and the findresource begin operator in the initialization section.

The %%EndResource comment defines the end of the file to parsers and spoolers.

The %%EOF comment formally signals the end of the file.

Referring to another CMap

If you want to create a CMap that differs from an existing CMap in only a few mappings, you can refer to the existing CMap instead of copying it. Let’s look at a CMap that refers to the previous one. The base CMap is for a horizontal Japanese font. This one is for the corresponding vertical Japanese font, which differs from the horizontal font only in the characters sensitive to writing direction. For example, the vertical font has top and bottom parentheses instead of left and right parentheses.

The comment section contains a line that refers to the base CMap:

```
%%IncludeResource: CMap CS-2093-H
```

The identification section contains a usecmap operator:

```
/CS-2093-H usecmap
```

This operator tells the PostScript interpreter to use the mappings in CS-2093-H, except where this CMap overrides them.

The mapping section contains only the mappings that differ from the base CMap.

CIDFonts

A CIDFont is a PostScript font resource that contains character descriptions identified by CID.

A CIDFont has two parts. The first part is a PostScript program that defines a dictionary object called a CIDFont resource. This part resides on the host.

The second part contains the actual character descriptions, the data that determines the appearance of printed characters. This part can reside either on the host or on disk, tape, or CD-ROM.

Adding CIDFonts

Type Transformer locates CIDFonts by the path name in the ATMDATA.DAT file. You can use **Edit ATMDATA.DAT**in the **Options** menu to change the path name.

Part 1: PostScript program

Part 1 of a CIDFont contains the following sections:

- Header comments
- Initialization
- Identification
- Font box
- More identification
- Other PostScript information

Header comments

To ensure correct processing in all environments, every CIDFont file must begin with the comment characters %! (percent sign and exclamation point). Comment lines after the first begin with the characters %% (two percent signs).

A typical CIDFont comment section looks like this:

```
%!PS-Adobe-3.0 Resource-CIDFont
%%DocumentNeededResources: ProcSet CIDInit
%%IncludeResource: ProcSet CIDInit
%%BeginResource: CIDFont (HeiseiMinchoW3)
%%Title: (HeiseiMinchoW3 IBM Japan2 0)
%%Version: 1
```

This comment:

```
%!PS-Adobe-3.0 Resource-CIDFont
```

identifies the file as a CIDFont that conforms to the conventions of PostScript Version 3.0.

These comments:

```
%%DocumentNeededResources: ProcSet CIDInit
%%IncludeResource: ProcSet CIDInit
```

identify a system support file, called CIDInit, to spoolers and parsers. VM uses this file to process CIDFonts. The %%DocumentNeededResources comment indicates that CIDInit is a required external resource. The %%IncludeResource comment indicates that if CIDInit is not available in the PostScript interpreter, it should be included inline.

This comment:

```
%%BeginResource: CIDFont (HeiseiMinchoW3)
```

informs spoolers and parsers that this file is a CIDFont named HeiseiMinchoW3. There is a corresponding %%EndResource comment at the end of the file.

This comment:

```
%%Title: (HeiseiMinchoW3 IBM Japan2 0)
```

states the CIDFont name (HeiseiMinchoW3), registry name (IBM), ordering name (Japan2), and supplement number (0).

This comment:

```
%%Version: 1
```

states the version number. This is the same as the version number in the identification section of the CIDFont.

Initializing the CIDFont

The next two lines initialize the CIDFont:

```
/CIDInit /Procset findresource begin
20 dict begin
```

The findresource begin operator directs the system to find the CIDInit file. There is a matching end operator near the end of the CIDFont file.

The begin operator defines a PostScript language dictionary and pushes it onto the dictionary stack. A CIDFont is a PostScript dictionary.

Identifying the CIDFont

The next section identifies the character collection on which the CIDFont is based, the name of the CIDFont, and the version and type of the CIDFont:

```
/CIDFontName  /HeiseiMinchoW3 def
/CIDFontVersion 1 def
/CIDFontType 0 def

/CIDSystemInfo 3 dict  dup begin
  /Registry (IBM) def
  /Ordering (Japan2) def
  /Supplement 0 def
end def
```

The CIDFontName line gives the name of the CIDFont. This is the same name as in the header comments.

The CIDFontVersion and CIDFontType lines identify the version and type of the CIDFont. The version is the same version as in the header comments. The type for the CIDFonts described in this publication is always 0.

CIDFontName and CIDFontType are required. CIDFontVersion is optional.

The CIDSystemInfo group of def operators identifies the character collection: Japan2, created by IBM, initial ordering. The character collection must be compatible with the character collection on which the CIDFont is based.

Font bounded box

The next section contains one required line defining a bounded box large enough to enclose any of the characters in the CIDFont. The array of four numbers gives the coordinates of the box corners, at 1000 units per em.

```
/FontBBox [-26 -200 1000 900] def
```

For more information about FontBBox, see the *PostScript Language Reference Manual*.

Another identification number

The XUID is an array of numbers that together form a unique identifier for the CIDFont:

```
/XUID [116 11 10108] def
```

The first number identifies the organization that created the CMap: 116 is IBM.

XUID is optional.

Other PostScript information

The rest of Part 1 may contain other information for human readers or for PostScript programs using the font. This information is not read by the PostScript interpreter.

This example shows a FontInfo dictionary:

```
/FontInfo 2 dict dup begin
  /Notice ((c) Copyright IBM 2002. All rights reserved.) def
  /FullName (HeiseiMinchoW3) def
end readonly def
```

For an explanation of the entries in a FontInfo dictionary, or for other entries that you may see in Part 1 of a CIDFont, see the *PostScript Language Reference Manual*.

Part 2: Data section

The data section of a CIDFont begins with a %%BeginData comment and a StartData operator and ends with an %%EndData comment:

```
%%BeginData: 3437606 Binary Bytes
(Binary) 3437579 StartData
   ⋮
%%EndData
```

The comments indicate the beginning and end of the data section to parsers and spoolers. The BeginData comment also indicates whether the data is in ASCII or binary format, and how many bytes of data there are. The StartData operator indicates the data format and number of bytes to PostScript.

Between the StartData operator and the %%EndData comment, the data section contains four blocks:

- A CIDMap contains information about the location of each character description in the CIDFont and the corresponding font dictionaries.
- One or more SubrMaps contain information about the location of the subroutines used by the character descriptions.
- One or more subroutines may be used by the character descriptions.
- The character descriptions, or glyph data, determine the appearance of the printed characters.

Closing the CIDFont

The CIDFont ends with these two comments:

```
%%EndResource
%%EOF
```

The %%EndResource comment defines the end of the file to parsers and spoolers.

The %%EOF comment formally signals the end of the file.

How CMaps and CIDFonts work together

An Adobe CID-keyed font includes one CMap and one CIDFont. You can create many different CID-keyed fonts by pairing the same CMap with different CIDFonts, or the other way around.

Type Transformer differs a little from Adobe practice in this respect. A Type Transformer batch file can match a single CIDFont to several CMaps at once:

- One CMap containing horizontal double-byte characters and double-byte characters that are not sensitive to writing direction (the *horizontal CMap*). This is the only required CMap.
- One CMap containing vertical double-byte characters (the *vertical CMap*)
- Several CMaps containing single-byte characters: katakana, half-width characters, and romaji

Adobe practice, by contrast, would be to create two related fonts, one horizontal and one vertical. The horizontal font would have one CMap containing single-byte characters, double-byte characters that are not sensitive to writing direction, and horizontal characters. The vertical font would have one CMap referring to the horizontal font’s CMap and replacing only the SWD characters (see “Referring to another CMap” on page 8).

Character collection compatibility

Both parts of a CID-keyed font, the CMap and the CIDFont, must use compatible character collections. Every ordered character collection is uniquely identified by a combination of three identifiers:

- The *registry name* identifies the company that issued the ordering, for example, IBM.
- The *ordering name* identifies an ordered character collection, for example, Japan2.
- The *supplement number* identifies the level of the ordering. A supplement number of 0 means that this is the initial ordering; 1 means that new characters have been added to supplement 0, beginning with the next available CID, and so on.

The registry and ordering in the version control sections of the CIDFont and CMap must match. The supplement may vary, but if it does, some character codes may map to a CID of 0.

Keeping the same CMap and changing the CIDFont

You can use a single CMap with different CIDFont resource files to obtain different glyphs for the same character set. For example, you can associate a CMap for IBM's Japan2 ordered character collection with Heisei Mincho, Heisei Gothic, and Heisei Maru Gothic CIDFont resources.

Each pairing of one CMap and one CIDFont is a distinct CID-keyed font.

Figure 2 shows characters from two CMaps, each associated with two different CIDFonts. In a given CMap, the same character code is always mapped to the same CID and always indicates the same character, but changing the CIDFont changes the glyph printed to represent that character.

Code	CMap 1 83pv-RKSJ-H			CMap 2 Ext-RKSJ-V		
	CID	CIDFont 1 and 2		CID	CIDFont 1 and 2	
<82AB>	851	お	お	851	お	お
<57>	56	W	W	296	W	W
<8179>	690	【	【	7915	︻	︻
<817A>	691	】	】	7916	︼	︼
<8D7B>	2030	砿	砿	5853	礦	礦
<E1E6>	5853	礦	礦	2030	砿	砿
<92CD>	3051	掴	掴	7747	摑	摑
<81F6>	777	‡	‡	0		

Figure 2. One CMap and two CIDFonts or two CMaps and one CIDFont. This figure was created by Adobe Systems, Inc.

Keeping the same CIDFont and changing the CMap

You can use different CMaps with the same CIDFont resource. Here are some cases when you might want to do this:

- Different platforms have different system-specific character sets. The CMap for each platform contains only the characters in that platform's character set; while the CIDFont is based on a character collection that is a union of all the character sets. The common CIDFont makes the CID-keyed font portable from one platform to another.
- You need variations of a font. For example, you need a horizontal and a vertical version of a Japanese font. In this case, the horizontal and vertical CMaps are identical except for characters that are sensitive to writing direction.

  Note: Type Transformer treats such a pair of horizontal and vertical CMaps and one CIDFont as a single font. In Adobe terminology, they are two separate fonts.

Figure 2 on page 12 shows glyphs from two CIDFont resources, each associated with two CMaps. In a given CIDFont, the same CID is always mapped to the same character descriptions, but changing the CMap means that the CID may no longer map to the same character code. Figure 2 on page 12 shows some differences in mapping:

- Character code 82A8 is mapped to the same CID in CMap1 and CMap2.
- Character code 57 is mapped to a proportionally spaced Roman character in CMap1 and to a half-width Roman character in CMap1. The two CMaps are intended for different platforms. One platform does not support half-width characters.
- Character codes 8179 and 817A are mapped to left and right parentheses in CMap1, for a horizontal font, and to top and bottom parentheses in CMap2, for a vertical font.
- Character codes 8D7B and E1E6 show character swapping. One CMap places the new version of the character in the primary code position (the one used most often) and maintains the old version in a secondary position, the other makes the old version primary and places the new version in secondary position.
- Character code 92CD maps to different characters in each CMap.
- Character code 81F6 is mapped to the dagger symbol in CMap1 and to the default notdef character in CMap2. CMap2 may be based on an earlier character collection that does not include the dagger symbol.

Putting the pieces together

Adobe associates a CMap and CIDFont to create a CID-keyed font by giving the font a name that includes the name of both component files, joined with a single or, preferably, double hyphen. The PostScript findfont operator locates the CMap and CIDFont and puts them togther.

Type Transformer associates one or more CMaps and a CIDFont by naming them in the same batch file (see the sample files on the *IBM Infoprint Fonts: Type Transformer and Utilities for Windows*® CD-ROM.

Rearranged fonts

A *rearranged font* combines characters from two or more other fonts. The most common reason to create one is to add user-defined characters to a font.

Figure 3 on page 14 shows an example of a CMap for a rearranged font. This font adds user-defined characters to Japanese Heisei Mincho.

```
%!PS-Adobe-3.0 Resource-Font
%ADOResourceSubCategory: RearrangedFont
%%DocumentNeededResources: procset CIDInit
%%+ font (IBDH0300)
%%IncludeResource: procset CIDInit (IBDH0300)
%%BeginResource: (JHMNUDCH)
%%Version: 1

/CIDInit /ProcSet findresource
begin
  %ADOStartRearrangedFont
  /JHMNUDCH

  [ /CS-2093-H
    /HeiseiMinchoUDC42Horiz
    /HeiseiMinchoUDC69Lo
    /HeiseiMinchoUDC69Hi
  ] beginrearranged font

    0 usefont
   11 begincidrange
      <6a41> <6a4b> 8009
      <6a4c> <6a60> 8038
      <6a61> <6a7c> 8138
      <6a7d> <6a96> 8197
      <6a97> <6a9f> 8286
      <6aa0> <6ab4> 7560
      <6ab5> <6ab5> 8091
      <6ab6> <6abf> 8102
      <6ac0> <6acf> 7585
      <6ad0> <6adf> 7940
      <6ae0> <6afe> 8321
      endcidrange

    1 usefont
    2 beginbfrange
      <424d> <424d> 44
      <425d> <425d> 60
      endbfrange

    2 usefont
    1 beginbfrange
      <6941> <699f> 32
      endbfrange

    3 usefont
    1 beginbfrange
      <69a0> <69fe> 32
      endbfrange

  endrearranged font
end
%%EndResource
%%EOF
```

Figure 3. CMap for a rearranged font: JHMNUDCH.CMP

Comments

The comment lines:

```
%!PS-Adobe-3.0 Resource-Font
%ADOResourceSubCategory: RearrangedFont
%%DocumentNeededResources: procset CIDInit
%%+ font (IBDH0300)
%%IncludeResource: procset CIDInit (IBDH0300)
%%BeginResource: (JHMNUDCH)
%%Version: 1
```

indicate that this is a PostScript font file, specifically a rearranged font, tell spoolers and parsers to make sure that the font file IBDH0300 is available, and tell spoolers and parsers that the name of this file is JHMNUDCH.

Component font files

These lines:

```
  [ /CS-2093-H
    /HeiseiMinchoUDC42Horiz
    /HeiseiMinchoUDC69Lo
    /HeiseiMinchoUDC69Hi
  ] beginrearranged font
```

name the fonts that contribute to this rearranged font file. The first font is the template: it is the basis for the rearranged font. Characters from the other fonts are added to the characters in the template or replace characters in the template.

In the rest of the file, the fonts will be referred to by number in the order they are named, starting with 0:

0	CS-2093-H
1	HeiseiMinchoUDC42Horiz
2	HeiseiMinchoUDC69Lo
3	HeiseiMinchoUDC69Hi

Only CS-2093-H is named in the comments. It is the only complete font file. The others are skeleton files containing only the UDCs you created.

Mapping the base font

This line:

```
    0 usefont
```

indicates that the next lines refer to font 0, IBDH0300.0

These lines:

```
   11 begincidrange
      <6a41> <6a4b> 8009
      <6a4c> <6a60> 8038
      <6a61> <6a7c> 8138
      <6a7d> <6a96> 8197
      <6a97> <6a9f> 8286
      <6aa0> <6ab4> 7560
      <6ab5> <6ab5> 8091
      <6ab6> <6abf> 8102
      <6ac0> <6acf> 7585
      <6ad0> <6adf> 7940
      <6ae0> <6afe> 8321
      endcidrange
```

are the same kind of mapping section as in a regular CMap (see “Mapping information” on page 6). They name the character codes from IBDH0300 that are used in the rearranged font and map them to CIDs.

Adding user-defined characters

These lines:

```
1 usefont
2 beginbfrange
  <424d> <424d> 44
  <425d> <425d> 60
  endbfrange
```

indicate that the rearranged font is now using font 1, HeiseiMinchoUDC42Horiz (Heisei Mincho, user-defined characters from code page section 42 for horizontal writing). Characters starting at character code 424D in IDBH0300, and continuing through 424D (one character), are replaced with the character from HeiseiMinchoUDC42Horiz whose decimal encoding is 44 (equivalent to X'2C'). In the same way, the characters from 425D to 425D (again, one character) in IDBH0300 are replaced with the character from HeiseiMinchoUDC42Horiz whose decimal encoding is 60 (equivalent to X'2C').

You derive the hexadecimal encoding from the second byte of the character code like this:

- For low second bytes (X'9F' or less), subtract X'21' from the second byte. For example:

  ```
  X'41' − X'21' = X'20'
  X'8C' − X'21' = X'6B'
  ```

- For high second bytes (X'A0' or greater), subtract X'80' from the second byte. For example:

  ```
  X'A0' − X'80' = X'20'
  X'DD' − X'80' = X'5D'
  ```

If you use the decimal equivalent of the hexadecimal encoding, omit the angle brackets around it.

Chapter 3. Creating user-defined characters

Type Transformer allows you to create user-defined characters (UDCs). UDCs are created in Type 1 outline format.

You can create UDCs in two places:

- In the sections containing characters that are sensitive to writing direction (SWD). For all languages, these are sections 41–44. These sections also contain resident characters.
- In the sections reserved for UDCs. These sections vary by language:

  Table 2. DBCS font sections reserved for UDCs

Language	**Sections reserved for UDCs**
Japanese	69–89
Korean	D4–DD
Simplified Chinese	76–80
Traditional Chinese	C2–E2

  Any characters you create in these sections must not be SWD.

You make your UDCs accessible for printing by creating a rearranged font CMap file. Because rearranged font CMaps can include characters in both CID-keyed and Type 1 formats, you can add Type 1 UDCs to a CID-keyed font.

To add a user-defined character to a font, use the procedure in this chapter as a guideline. You may create many UDCs at a time, but for simplicity, this example shows the creation of a single UDC.

You will perform the following steps:

1. Decide where to add the UDC
2. Design the new UDC
   a. Prepare to use FontLab
   b. Design the UDC
   c. Finish in FontLab
3. Create and modify a UDC code page
4. Modify the rearranged CMap file
5. Check the ATMDATA.DAT file
6. Create or modify a Type Transformer batch file
7. Run a Type Transformer DB
8. Modify the Coded Font (simulation fonts only)
9. Install font objects in your font library

Note: If a UDC already exists in raster format, and you want to add the same UDC to an outline font, you may be able to convert the existing UDC instead of recreating it. Follow the instructions in Chapter 4, “Converting AFP raster UDCs to Type 1 outline UDCs” on page 41, then return to this chapter at “Creating and modifying the UDC code page” on page 30.

Deciding where to add the UDC

Table 3 shows how DBCS font sections are used: which sections contain resident characters, which sections contain SWD characters, and which sections are available for UDCs.

Table 3. How DBCS font sections are used

Language	Resident font characters; SWD; UDCs allowed	Resident font characters; not SWD; UDCs not allowed	No resident font characters; not SWD; UDCs allowed
Japanese	41–44	45–68, B3–DA, DC–EC	69–89
Korean	41–44	45–4E, 50–6C, 84–D3	D4–DD
Simplified Chinese (GB)	41–44	45-46, 48-6C	76–7F
Simplified Chinese (GB18030)	41–44	45–75, 81–EF, F6–F8, FB–FC	76–80
Traditional Chinese	41–44	45–49, 4C–91	C2–E2

Within a section where UDCs are allowed, select the character code where you want to add the UDC. This may be either an unused character code or a character you want to replace. If you are adding a new UDC at an unused character code, select the lowest value available.

Character codes are the same as the code points in the primary code page used with the font. Table 4 shows the character codes you can use; and the following publications show code pages:
- *IBM Infoprint Fonts: Japanese Font Library Technical Reference*, S544-5849
- *IBM Infoprint Fonts: Korean Font Library Technical Reference*, S544-5850
- *IBM Infoprint Fonts: Simplified Chinese Font Library Technical Reference*, S544-5851
- *IBM Infoprint Fonts: Traditional Chinese Font Library Technical Reference*, S544-5852

Table 4. Character codes where you can create UDCs

Language	First byte (section)	Second byte (low)	Second byte (high)
Japanese	X'41'–X'44'	X'41'–X'9F'	X'A0'–X'FE'
	X'69'–X'88'	X'41'–X'9F'	X'A0'–X'FE'
	X'89'	X'41'–X'9F'	X'A0'–X'BD'
Korean	X'41'–X'44'	X'41'–X'9F'	X'A0'–X'FE'
	X'D4'–X'DD'	X'41'–X'7F' X'81'–X'9F'	X'A0'–X'FD'
Simplified Chinese (GB)	X'41'–X'44'	X'41'–X'9F'	X'A0'–X'FE'
	X'76'–X'7F'	X'41'–X'7F' X'81'–X'9F'	X'A0'–X'FD'
Simplified Chinese (GB18030)	X'41'–X'44'	X'41'–X'9F'	X'A0'–X'FE'
	X'76'–X'7F'	X'41'–X'7F' X'81'–X'9F'	X'A0'–X'FD'
	X'80'	X'41'–X'7F' X'81'–X'9F'	
Traditional Chinese	X'41'–X'44'	X'41'–X'9F'	X'A0'–X'FE'
	X'C2'–X'E2'	X'41'–X'7F'X'81'–X'9F'	X'A0'–X'FD'

Characters whose second byte is in the range of X'41'–X'9F' are called “low” characters. Characters whose second byte is in the range of X'A0'–X'FE' are called “high” characters. It is important to know whether a character is low or high when selecting FontLab objects.

Designing the new UDC

The instructions in this section explain how to use FontLab to design a new UDC. If the UDC already exists in raster format, and you want to add the same UDC to an outline font, you may be able to convert the existing UDC instead of recreating it. Follow the instructions in Chapter 4, “Converting AFP raster UDCs to Type 1 outline UDCs” on page 41, then return to this chapter at “Creating and modifying the UDC code page” on page 30.

Preparing to use FontLab

1. Determine the name of the FontLab file where you will create the UDC, and the alias name of the font (the Type 1 full name as it will appear in the rearranged font CMap):

Table 5. FontLab file names and alias names

Font	FontLab VFB file name	Alias name
Japanese Heisei Kaku Gothic	JHKG*dssn*.VFB	HeiseiKakuGothicUDC*ssdddddddnn* HeiseiKakuGothicUDC*ssnn*
Japanese Heisei Maru Gothic	JHMG*dssn*.VFB	HeiseiMaruGothicUDC*ssdddddddnn* HeiseiMaruGothicUDC*ssnn*
Japanese Heisei Mincho	JHMN*dssn*.VFB	HeiseiMinchoUDC*ssdddddddnn* HeiseiMinchoUDC*ssnn*
Korean Gothic	HKG2*dssn*.VFB	GothicKG2UDC*ssdddddddnn* GothicKG2UDC*ssnn*
Korean Myengjo	HSM2*dssn*.VFB	MyengjoSM2UDC*ssdddddddnn* MyengjoSM2UDC*ssnn*
Simplified Chinese Fang Song	SFSG*dssn*.VFB	FangSongSCUDC*ssdddddddnn* FangSongSCUDC*ssnn*
Simplified Chinese Hei	SHEI*dssn*.VFB	HeiSCUDC*ssdddddddnn* HeiSCUDC*ssnn*
Simplified Chinese Kai	SKAI*dssn*.VFB	KaiSCUDC*ssdddddddnn* KaiSCUDC*ssnn*
Simplified Chinese Song	SSNG*dssn*.VFB	SongSCUDC*ssdddddddnn* SongSCUDC*ssnn*
Traditional Chinese Kai	TKAI*dssn*.VFB	KaiTCUDC*ssdddddddnn* KaiTCUDC*ssnn*
Traditional Chinese Sung	TSNG*dssn*.VFB	SungTCUDC*ssdddddddnn* SungTCUDC*ssnn*

where:

d in the FontLab file name indicates the writing direction: H (horizontal), V (vertical), or B (both).

dddddd
in the alias name for sections 41–44 indicates the writing direction: Horiz or Vert. The alias name for sections 45 and up does not indicate the writing direction, because no SWD characters are allowed in these sections.

ss in both names is the first byte of the character code (the same as the code page section number).

n in the FontLab file name is L for low second bytes (X'9F' or less), or H for high second bytes (X'A0' or greater).

nn in the alias name is Lo for low second bytes or Hi for high second bytes.

Table 6 shows an example for each language. Note that both horizontal and vertical versions of SWD characters are being created.

Table 6. Examples of FontLab file names

Language	**Typeface**	**Character code**	**Writing direction**	**FontLab file name**	**Alias name**
Japanese	Heisei Gothic	4142	Horizontal	JHKGH41L.VFB	HeiseiKakuGothicUDC41HorizLo
			Vertical	JHKGV41L.VFB	HeiseiKakuGothicUDC41VertLo
Korean	Myengjo	D5AB	Not SWD	HSM2BD5H.VFB	MyengjoSM2UDCD5Hi
Simplified Chinese	Fang Song	7650	Not SWD	SFSGB76L.VFB	FangSongSCUDC76Lo
Traditional Chinese	Kai	43A0	Horizontal	TKAIH43H.VFB	KaiTCUDC43HorizHi
			Vertical	TKAIV43V.VFB	KaiTCUDC43VertHi

2. Determine the FontLab encoding for the character.
   - For low characters (second byte X'9F' or less), subtract X'21' from the second byte, then convert the result to decimal. For example:
     ```
     X'41'–X'21' = X'20' =  32
     X'8C'–X'21' = X'6B' = 107
     ```
   - For high characters (second byte X'A0' or greater), subtract X'80' from the second byte, then convert the result to decimal. For example:
     ```
     X'A0'–X'80' = X'20' =  32
     X'DD'–X'80' = X'5D' =  93
     ```

Designing the UDC

The following steps show how you might design a character. For complete instructions, see *FontLab User's Guide*.

1. **Decide if any other characters can be used as a starting point.**

   Figure 4 on page 21 shows what you want to do in this example. You will combine the 4562 and 46AA characters from IBDH0300 to form a new Japanese character, in the Heisei Mincho typeface. (4562 and 46AA are hexadecimal character codes.)

   In fact, a character that looks like the desired result already exists at character code 45C1. You will create a duplicate for practice at character code 6941, the lowest character code available for Japanese UDCs that are not sensitive to writing direction.

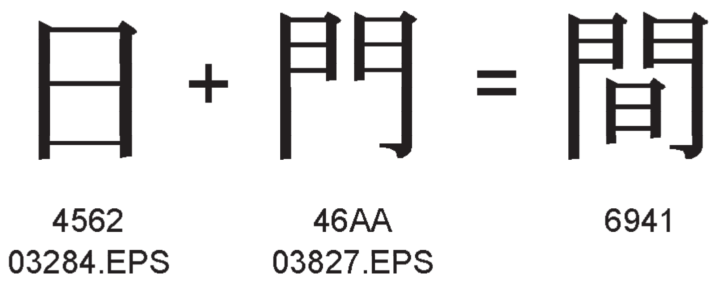


Figure 4. Combining two existing characters to make a UDC. Where 4562, 46AA, and 6941 are code points, and 03284 and 03827 are CID numbers.

You need to create EPS images of 4562 and 46AA so that they can be imported into FontLab at the appropriate location. You do this by using CID2EPS (see *IBM Infoprint Fonts: Introduction to Type Transformer and Utilities for Windows*) to extract images of the characters from a font.

You must determine:

- The CIDFont file name.

  The IBM-supplied CIDFont files for DBCS fonts are listed in *IBM Infoprint Fonts: Font Summary*. The CIDFont file for Japanese Heisei Mincho is IBJHMNW3.CID.
- The CMap name.
  - If you know the character code (the code point) for the character you want to extract, use the IBM-supplied CMap associated with that code page. These CMaps are listed in *IBM Infoprint Fonts: Font Summary*. Use the horizontal CMap for horizontal characters and characters that are not sensitive to writing direction. Use the vertical CMap for vertical characters. In this example, for a non-SWD Japanese character, you can use IBDH0300.CMP.
  - If you know the character identifier (CID), use one of the special IBM-supplied CMaps that contain all the characters for one language:

Language	**CMap**
Japanese	IBJPNALL.CMP
Korean	IBKORALL.CMP
Simplified Chinese (GB)	IBCHSALL.CMP
Simplified Chinese (GB18030)	ILCHSALL.CMP
Traditional Chinese	IBCHTALL.CMP

    In these special CMaps, the character code is always the same as the CID.
  - If you don't know either the character code or the CID, look through one of the following until you find an image of the character:
    - *IBM Infoprint Fonts: Japanese Font Library Technical Reference*
    - *IBM Infoprint Fonts: Korean Font Library Technical Reference*
    - *IBM Infoprint Fonts: Simplified Chinese Font Library Technical Reference*
    - *IBM Infoprint Fonts: Traditional Chinese Font Library Technical Reference*

If you find the character in a code page, the character code in the CMap associated with that code page is the same as the code point. If you find the character in the "Characters" chapter, the number printed under it is the CID.

Use CID2EPS to create the image of a character with the following parameter (see *IBM Infoprint Fonts: Introduction to Type Transformer and Utilities for Windows*).

Either of these parameters creates an image of character code 4562 (CID 3284):

```
CIDFont Name: IBJHMNW3.CID
CMap Name: IBDH0300.CMP
Code: 4562 (hexadecimal)

CIDFont Name: IBJHMNW3.CID
CMap Name: IBJPNALL.CMP
Code: 3284 (decimal)
```

Either of these parameters creates an image of 46AA (CID 3827):

```
CIDFont Name: IBJHMNW3.CID
CMap Name: IBDH0300.CMP
Code: 46AA (hexadecimal)

CIDFont Name: IBJHMNW3.CID
CMap Name: IBJPNALL.CMP
Code: 3927 (decimal)
```

CID2EPS creates an image called 03284.EPS and 03827.EPS in this example.

If you extracted the character by character code, you may not know the CID. To determine the CID, find the character code (code point) in the code page images in one of the following:
- *IBM Infoprint Fonts: Japanese Font Library Technical Reference*
- *IBM Infoprint Fonts: Korean Font Library Technical Reference*
- *IBM Infoprint Fonts: Simplified Chinese Font Library Technical Reference*
- *IBM Infoprint Fonts: Traditional Chinese Font Library Technical Reference*

The alphanumeric identifier under the character is the graphic character global identifier (GCGID). The mapping tables in the online information on the *IBM Infoprint Fonts* CD-ROM map GCGIDs (in the third column) to CIDs (in the first column).

2. **Open the VFB file.**

VFB files are predefined Type 1 outlines in FontLab 3.0 internal format. They are stored in the *d:*\DUVTT*lang**xxxx* directory, where:

d is the drive.

lang
is CHS (Simplified Chinese), CHT (Traditional Chinese), JAPAN, or KOREA.

xxxx
indicates the typeface in both the directory name and the file name:

SFSG	Simplified Chinese Fang Song
SHEI	Simplified Chinese Hei
SKAI	Simplified Chinese Kai
SSNG	Simplified Chinese Song
TKAI	Traditional Chinese Kai
TSNG	Traditional Chinese Sung
JHKG	Japanese Heisei Kaku Gothic
JHMG	Japanese Heisei Maru Gothic
JHMN	Japanese Heisei Mincho
HKG2	Korean Gothic
HSM2	Korean Myengjo

For each section in which UDCs are permitted, there are two VFB files, Lo (code points up to 9F) and Hi (code points A0 and up).

FontLab uses the VFB files as input and creates PFB files, actual Type 1 outlines, as output. You will now add your character to a VFB input file.

Use Table 5 on page 19 to determine the name of the VFB file to open. In this example, it is JHMNB69L.VFB.

Start FontLab and select **File –> Open** to get to the VFB files. When you select JHMNB69L.VFB, a window like the one in Figure 5 appears.

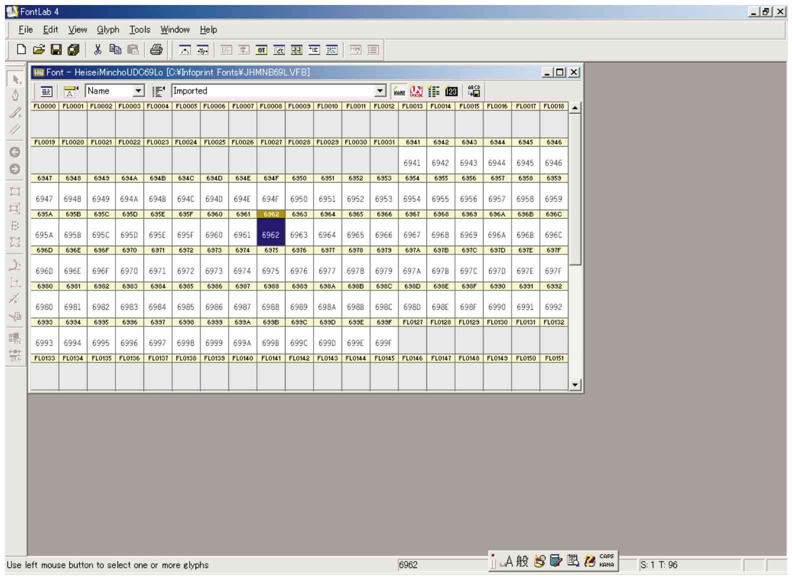
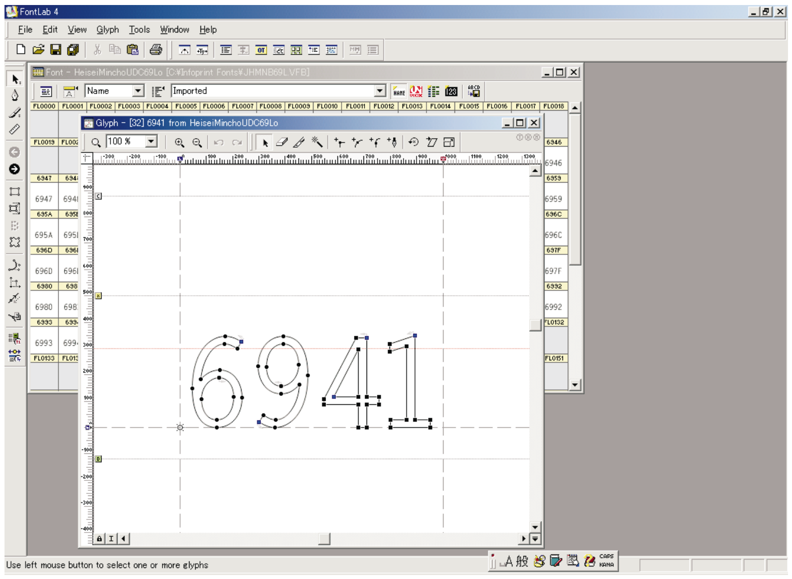

FontLab 4
File Edit View Glyph Tools Window Help
Font – HeiseiMinchoUDC69Lo [C:¥Infoprint Fonts¥JHMNB69L.VFB]
Name
Imported
Use left mouse button to select one or more glyphs
6962
S: 1 T: 96

Figure 5. Default view after opening a VFB file

3. **Move to the desired location.**

   Select the code point where you wish to add the character, then click the character on the Font window. The default window changes to show the code point you selected, and a window like Figure 6 on page 24 appears.

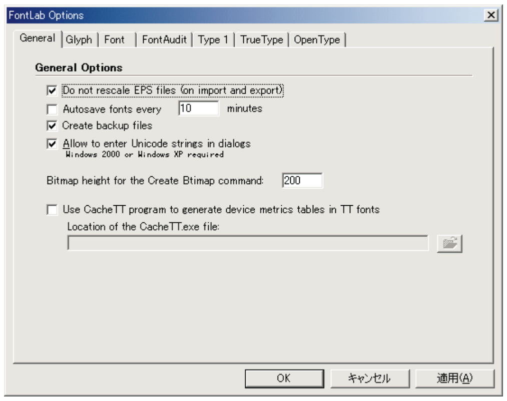
   In this example, select code point 6941, or FontLab encoding 32 in the lower part of section 69. (For how to derive 32 from 6941, see step 2 in “Preparing to use FontLab” on page 19.)


FontLab 4
File Edit View Glyph Tools Window Help
Glyph - [32] 6941 from HeiseiMinchoUDC69Lo
Name
Imported
100 %

Use left mouse button to select one or more glyphs

Figure 6. Glyph window of selected code point

4. **Delete the predefined glyph.** Select **Edit –> Select All...**, then **Edit –> Delete**. The predefined glyph is deleted.
5. **Check the Rescale parameter.** You must now make sure that the new character will not be rescales when you import it. Select **Tools –> Options** to open a window that looks like Figure 7.



General Options

- [✓] Do not rescale EPS files (on import and export)
- [ ] Autosave fonts every 10 minutes
- [✓] Create backup files
- [✓] Allow to enter Unicode strings in dialogs
  Windows 2000 or Windows XP required

Bitmap height for the Create Btimap command: 200

- [ ] Use CacheTT program to generate device metrics tables in TT fonts
  Location of the CacheTT.exe file:

Figure 7. Options window

Select the **General** tab and verify that the **Do not rescale EPS files (on import or export)** is checked.

6. **Paste one of the EPS files into the new location.**

   Select **Glyph –>Import from EPS...**. A file selection window is displayed. Select the 03827.EPS file. The window changes to look like Figure 8 on page 25.

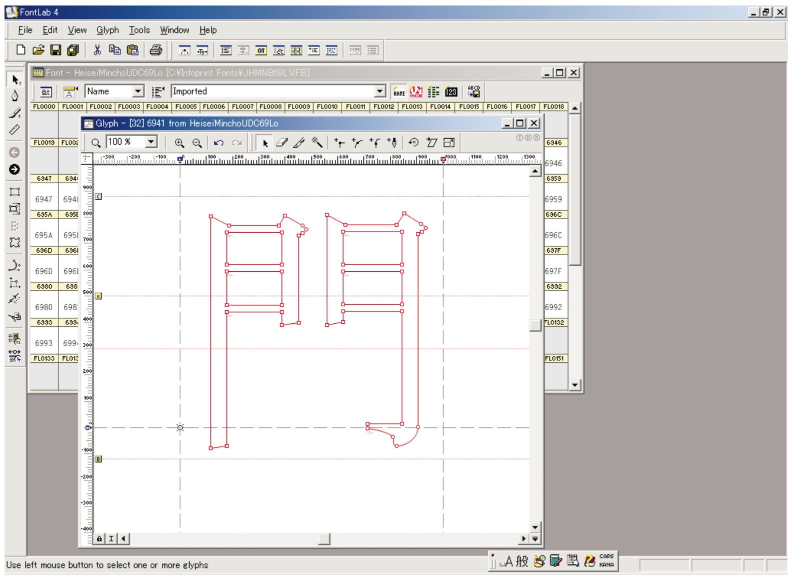

FontLab 4
File Edit View Glyph Tools Window Help
Font - HeiseiMinchoUDC69Lo
Glyph - [32] 6941 from HeiseiMinchoUDC69Lo
100 %
Name
Imported
Use left mouse button to select one or more glyphs

Figure 8. Character 6941. Character code 32 after 03827.EPS is pasted in.

7. **Draw guidelines for the maximum ascender and minimum descender values.**

   You must be careful that the finished new character does not extend above the maximum ascender value or below the minimum descender value in Table 7. If the character exceeds these values, when you print it in vertical text, it will overlap the characters above and below.

   Table 7. Maximum ascender and minimum descender values

Language	Ascender	Descender
Japanese	880	−120
Korean	880	−120
Simplified Chinese	880	−120
Traditional Chinese	880	−120

   Draw some guidelines to remind yourself where the limits are. Before drawing guidelines, you must set view options. Select **View –> Show Layers**, then check **Guidelines**.

   On the Tools palette, select **Edit** to put the mouse pointer in Edit mode. Go to the horizontal ruler at the top of the Edit window (under the window's title bar). Holding the left mouse button down, drag a horizontal guideline from the ruler until the Y coordinate in the lower left corner of the window matches the value of the maximum ascender. Draw another guideline for the minimum descender. You may need to change the magnification of the window to achieve the accuracy needed to place the guideline. To do this, select **View –> Zoom**.

   You must be careful that the finished new character does not extend metrics value 1,000. The guidelines of metrics are shown as gray dotted lines on Figure 8. It is not applied for the UDC in sections 41–44.

8. **Paste the second EPS file into a work area.**

   Move to a unassigned character position, which you will use as a work area for changing the second character before you add it to the first one. In all sections, high or low, character positions decimal 31 and below, and decimal 128 and above, are unassigned.

   To do this, double-click an unassigned character in the Font window twice, in this example decimal 16.

   Select **Glyph –> Import from EPS...**. A file selection window is displayed. Select the 03284.EPS file. The window changes to look like Figure 9.

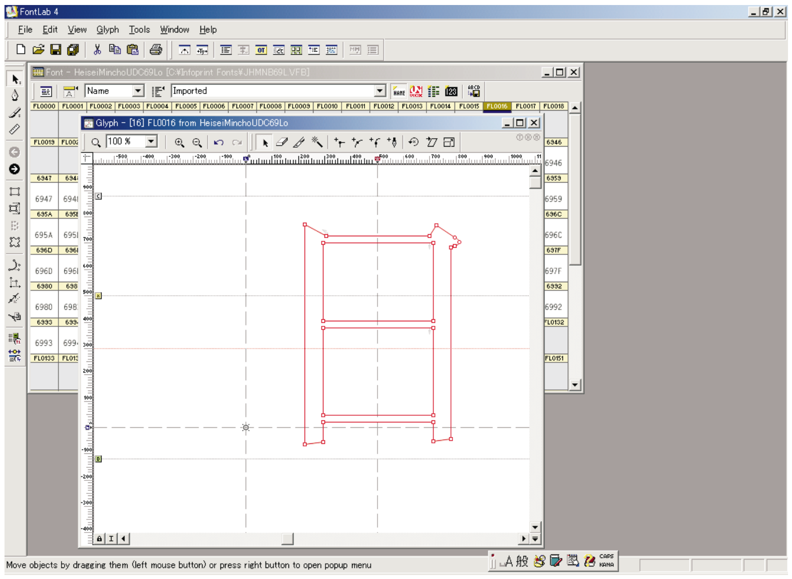

FontLab 4
File Edit View Glyph Tools Window Help
Font - HeiseiMinchoUDC69Lo [C:¥Infoprint Fonts¥JHMNB69L.VFB]
Name
Imported
Glyph - [16] FL0016 from HeiseiMinchoUDC69Lo
100 %
Move objects by dragging them (left mouse button) or press right button to open popup menu

Figure 9. Temporary character at decimal 16. Character code 16 after 03284.EPS is pasted in.

9. **Scale the image.**

   In this example, you have to scale the second image so that it is the right size to combine with the first image.

   Select **Tools –> Transform...**. A window like Figure 10 appears.

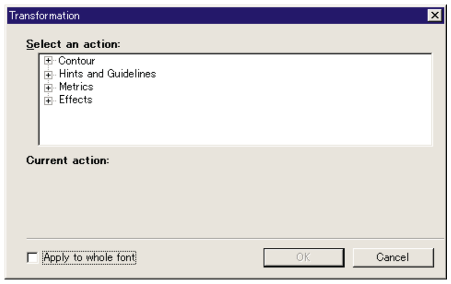
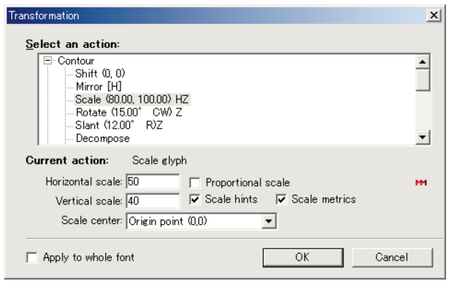


Figure 10. Dialog box for transform

   Select **Contour –> Scale...**. A window like Figure 11 on page 27 appears. Turn off **Proportional Scale** and scale the character 50% in the Horizontal direction and 40% in the vertical scale. These numbers were derived by measuring the character and estimating how much it needed to change to match the target. You can experiment until you are satisfied.


Transformation

Select an action:

- Contour
  - Shift (0, 0)
  - Mirror [H]
  - Scale (80.00, 100.00) HZ
  - Rotate (15.00° CW) Z
  - Slant (12.00° R)Z
  - Decompose

Current action: Scale glyph

Horizontal scale: 50 ☐ Proportional scale

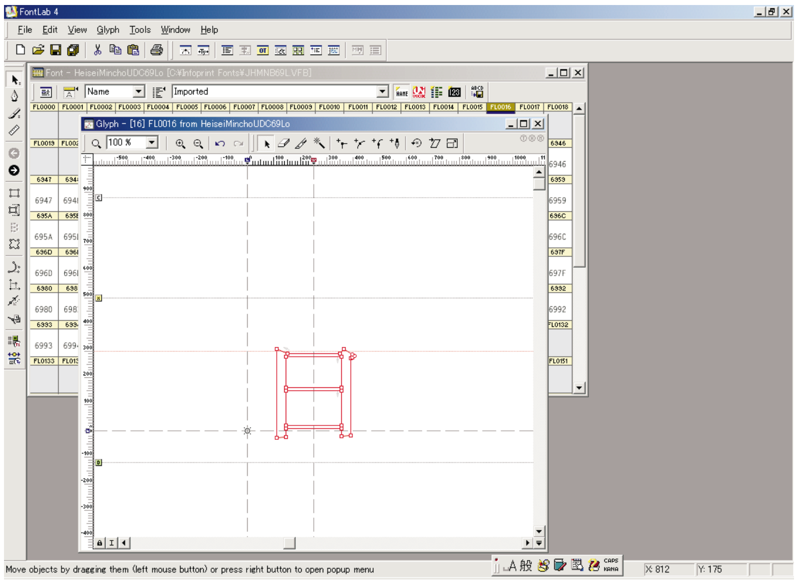
Vertical scale: 40 ☑ Scale hints ☑ Scale metrics

Scale center: Origin point (0,0)

☐ Apply to whole font

Figure 11. Dialog box used for scaling

After scaling is complete, the character looks like Figure 12.


FontLab 4
File Edit View Glyph Tools Window Help
Font - HeiseiMinchoUDC69Lo
Glyph - [16] FL0016 from HeiseiMinchoUDC69Lo
100 %
Name
Imported
X: 812
Y: 175

Move objects by dragging them (left mouse button) or press right button to open popup menu

Figure 12. Scaled character at decimal 16

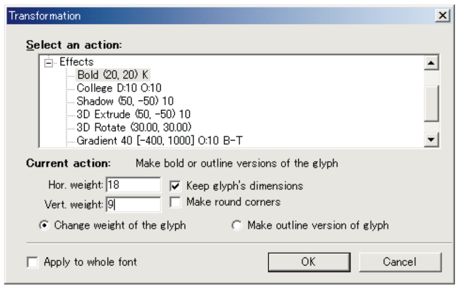
10. **Adjust the weight.**

    Because you scaled one character and not the other, the weights of the two characters are not equal. You must adjust the weight of the added character to match the weight of the original character.

    Select **Tools –> Transform**. A windows like Figure 10 on page 26 appears.

    Select **Effects –> Bold...**. A window like Figure 13 on page 28 appears. In this window, you can set different Horizontal and Vertical weights.

    To estimate the needed adjustments, click the right mouse button at points on either side of a stroke that you want to measure. X and Y coordinates for the point are shown in the lower right corner of the screen. In this example, try 18 for horizontal weight and 9 for vertical weight. You can experiment until you are satisfied.


Transformation

Select an action:

- Effects
  - Bold (20, 20) K
  - College D:10 O:10
  - Shadow (50, -50) 10
  - 3D Extrude (50, -50) 10
  - 3D Rotate (30.00, 30.00)
  - Gradient 40 [-400, 1000] O:10 B-T

Current action: Make bold or outline versions of the glyph

Hor. weight: 18 [✓] Keep glyph's dimensions

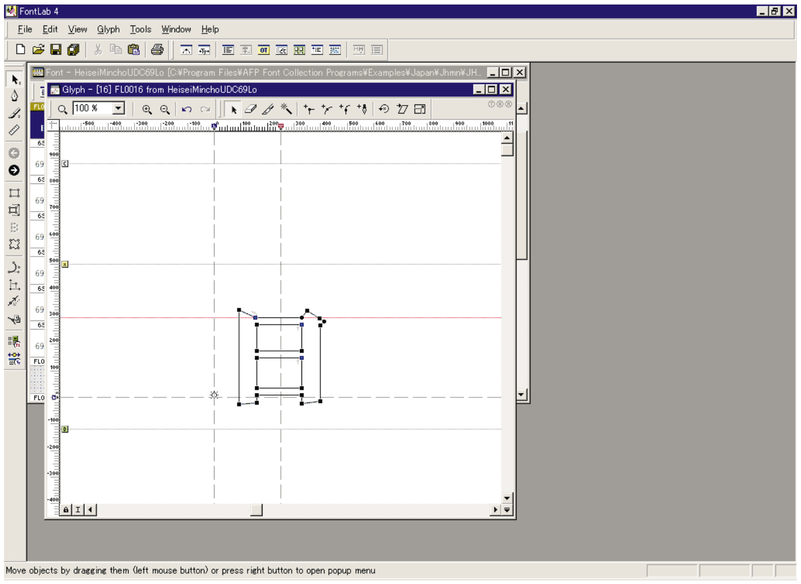
Vert. weight: 9 [ ] Make round corners

(●) Change weight of the glyph ( ) Make outline version of glyph

[ ] Apply to whole font

Figure 13. Make bold window

After you adjust the weight, you see a window like Figure 14.


FontLab 4
File
Edit
View
Glyph
Tools
Window
Help
Font – HeisеiMinchoUDC69Lo [C:¥Program Files¥AFP Font Collection Programs¥Examples¥Japan¥Jhmn¥JH...
Glyph – [16] FL0016 from HeiseiMinchoUDC69Lo
100 %

Move objects by dragging them (left mouse button) or press right button to open popup menu

Figure 14. After adjusting the weight

11. **Cut the character.**

    Select the complete character, then select **Edit –> Select All**. Then select **Edit –> Cut** to put the character on the clipboard.

12. **Move back to the new character.**

    Now move back to character 6941 by double-clicking a character in the Font window.

13. **Add the second character.**

    Add the cut character to the displayed character. Select **Edit –> Paste**. After adding the character, you see a window that looks like Figure 15 on page 29.

    The added character has been placed on top of the original character. The added character is still selected.

    Attention: Be careful not to select a point. If you do, you will lose your selection of the whole added character and you will not be able to move it.

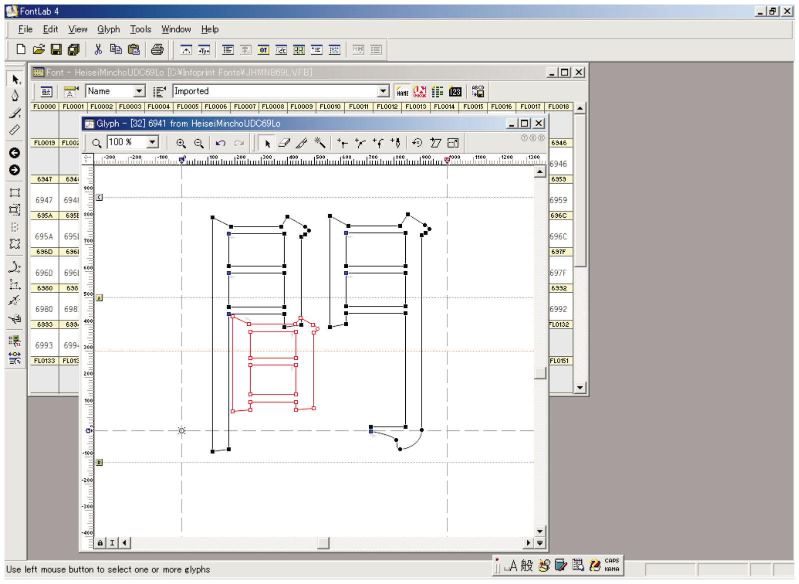

FontLab 4
File Edit View Glyph Tools Window Help
Font - HeiseiMinchoUDC69Lo [C:¥Infoprint Fonts¥JHMNB69L.VFB]
Glyph - [32] 6941 from HeiseiMinchoUDC69Lo
Use left mouse button to select one or more glyphs

Figure 15. After adding the second character to the first

14. **Move the added character.**

    The added character is not in the right position, so you must move it.

    Click on the contour of the added character and drag it to the right position.

    Attention: If you deselect the added character before you move it to the right position, you can undo the previous operation by clicking the **Undo** button in the Glyph window.

    The character is completed as shown in Figure 16.

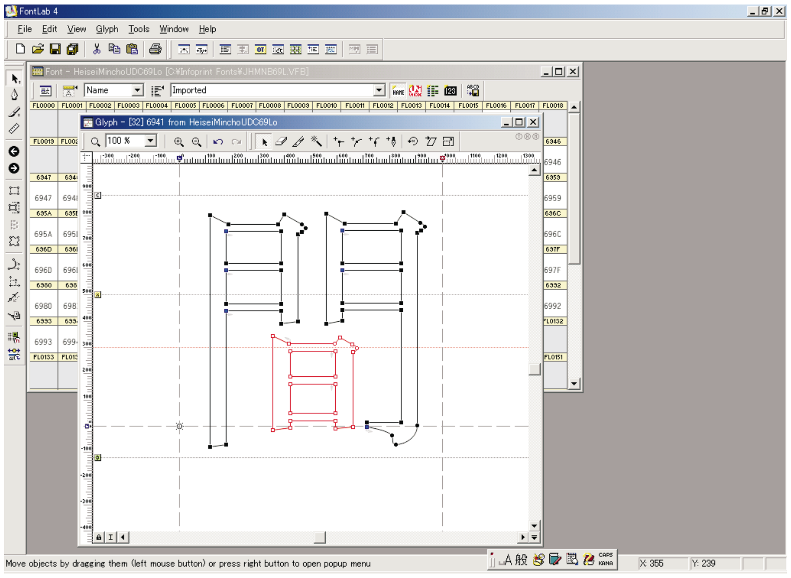

FontLab 4
File Edit View Glyph Tools Window Help
Font - HeiseiMinchoUDC69Lo [C:¥Infoprint Fonts¥JHMNB69L.VFB]
Glyph - [32] 6941 from HeiseiMinchoUDC69Lo
Move objects by dragging them (left mouse button) or press right button to open popup menu
X: 355
Y: 239

Figure 16. Completed character

15. **Clean up.**

    Now clean up the temporary work area in decimal character 16. Move to decimal character 16 in the Font window. It looks empty because you cut the character you had there to the clipboard, but it contains white space. Select **Edit –> Delete**.

Finishing in FontLab

Before you exit FontLab, you must check the Type 1 font parameters and export the new font.

1. Select **File –> FontInfo**. The Font Information window appears.

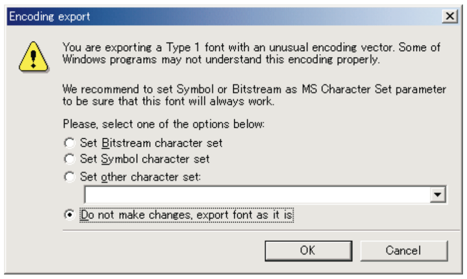
   Select **Metrics and Dimensions –> Key dimensions** and verify that:
   - The cap height is equal to the maximum ascender.
   - The X-height is approximately equal to half the cap height.

   Select the **Names and Copyright** tab and verify that the weight is appropriate for the font as follows:

Type 1 weight	**Font**
Medium	Japanese Heisei Kaku Gothic
SemiLight	Japanese Heisei Maru Gothic
Light	Japanese Heisei Mincho
Medium	Korean Gothic
Medium	Korean Myengjo
ExtraLight	Simplified Chinese Fang Song
SemiBold	Simplified Chinese Hei
Medium	Simplified Chinese Kai
Medium	Simplified Chinese Song
Medium	Traditional Chinese Kai
Light	Traditional Chinese Sung

   On the **Names and Copyright** tab, select **Copyright information** and verify that **Created by** is set to your company name.

   Select the **Version** tab and verify that the version number has been updated to indicate a change in the Type 1 font.
2. Export the new font. Select **File –> Generate font...**, then select **Type1 binary (*.pfb)** as the file type. The file name is the same as the VFB file you opened, in this example JHMNB69L, with an extension of PFB. This file is the actual new Type 1 font. When Figure 17 appears, check **Do not make changes, export font as it is**.



You are exporting a Type 1 font with an unusual encoding vector. Some of Windows programs may not understand this encoding properly.

We recommend to set Symbol or Bitstream as MS Character Set parameter to be sure that this font will always work.

Please, select one of the options below:

- Set Bitstream character set
- Set Symbol character set
- Set other character set:
- [✓ Do not make changes, export font as it is]

Figure 17. Encoding export window

Creating and modifying the UDC code page

You need to create a UDC code page in order to print UDCs.

1. To avoid overriding existing code pages, save all existing code pages by copying them to a new directory.
2. Copy the appropriate sample UDC code page on the *IBM Infoprint Fonts: Type Transformer and Utilities for Windows* CD-ROM to your code page directory.

   Give the copy the same name as the primary code page.

The sample UDC code pages are:
d:\AFPFONTS\CHS\T10837U
d:\AFPFONTS\CHS\T1K837U
d:\AFPFONTS\CHT\T10835U
d:\AFPFONTS\JAPAN\T10300U
d:\AFPFONTS\JAPAN\T1K300U
d:\AFPFONTS\KOREA\T10834U
d:\AFPFONTS\KOREA\T1K834U

The sample UDC code pages are predefined to refer to all possible code points.

3. Now that you have a UDC code page, you must modify it. Use the AFP Font Editor to convert the code page to a flat file (see *IBM Infoprint Fonts: Introduction to Type Transformer and Utilities for Windows*, G544-5853).
4. Edit the flat file and remove all code points except the one you want to use for your UDC.
5. Use the AFP Font Editor to reconvert the flat file to a code page.

Modifying the rearranged CMap file

You need to modify the appropriate rearranged font CMap file supplied with Type Transformer. The file name is *d*:\DUVTT\*lang**xxxx**xxxx*UDC*n*.CMP, where:

d is the drive.

lang
is CHS (Simplified Chinese), CHT (Traditional Chinese), JAPAN, or KOREA.

xxxx
indicates the typeface in both the directory name and the file name:

SFSG	Simplified Chinese Fang Song
SHEI	Simplified Chinese Hei
SKAI	Simplified Chinese Kai
SSNG	Simplified Chinese Song
TKAI	Traditional Chinese Kai
TSNG	Traditional Chinese Sung
JHKG	Japanese Heisei Kaku Gothic
JHMG	Japanese Heisei Maru Gothic
JHMN	Japanese Heisei Mincho
HKG2	Korean Gothic
HSM2	Korean Myengjo

n is the writing direction: H (horizontal) or V (vertical).

Note: If you are creating characters that are SWD, modify *both* the horizontal and vertical files. If you are creating characters that are not SWD, modify only the horizontal file.

Type 1 fonts for the sections where UDCs are permitted are identified in the CMap files. Comment flags (%, the percent sign) are used to cause sections *not* to be included in a rearranged font. You must make sure that any characters you added are included.

- **For SWD sections 41–44:**
  1. *Remove* the comment flag from the section you modified. For example, if you added two characters for horizontal writing to the lower part of section 42 of Korean Gothic, edit HKG2UDCH and remove the comment flags from the modified section *and* from all *preceding* sections. If you modified a low section, do not remove the comment flag from the corresponding high section. For example, change:

```
%ADOStartRearrangedFont
/HKG2UDCH
  [ /IBDH0834
%   /GothicKG2UDC41HorizLo
%   /GothicKG2UDC41HorizHi
%   /GothicKG2UDC42HorizLo
%   /GothicKG2UDC42HorizHi
%   /GothicKG2UDC43HorizLo
%   /GothicKG2UDC43HorizHi
%   /GothicKG2UDC44HorizLo
%   /GothicKG2UDC44HorizHi
```

     to:

```
%ADOStartRearrangedFont
/HKG2UDCH
  [ /IBDH0834
    /GothicKG2UDC41HorizLo
    /GothicKG2UDC41HorizHi
    /GothicKG2UDC42HorizLo
%   /GothicKG2UDC42HorizHi
%   /GothicKG2UDC43HorizLo
%   /GothicKG2UDC43HorizHi
%   /GothicKG2UDC44HorizLo
%   /GothicKG2UDC44HorizHi
```

  2. Add your character to the mapping section by creating an entry between `beginbfrange` and `endbfrange`. For example, if you added characters with decimal encodings 44 and 60 at 424D and 425D, change the existing mapping:

```
%-----------------------------------------------
% Section 42
%-----------------------------------------------
% 3 usefont
% 1 beginbfrange
%   <4241> <429f> 32
%   endbfrange
% 4 usefont
% 1 beginbfrange
%   <42a0> <42fe> 32
%   endbfrange
```

     to look like this:

```
%-----------------------------------------------
% Section 42
%-----------------------------------------------
  3 usefont
  2 beginbfrange
    <424d> <424d> 44
    <425d> <425d> 60
%   endbfrange
% 4 usefont
% 1 beginbfrange
%   <42a0> <42fe> 32
%   endbfrange
```

Do not simply remove the comment flags from the whole section. Note that because you modified two character code ranges, you must change the value at the beginning of the `beginbfrange` line from 1 to 2. Remember that angle brackets around a number indicate a hexadecimal value. The angle brackets around 44 and 60 are omitted because these are decimal encodings.

- **For reserved UDC sections other than 41–44:**
  1. *Add* a comment flag to every section *after* the one you modified, except that if you modified a low section, do not comment out the corresponding high section. For example, if you added a character to the lower part of section 69 of Japanese Heisei Mincho, change:

```
%ADOStartRearrangedFont
/JHMNUDCH
  [ /CS-K2093-H
    /HeiseiMinchoUDC69Lo
    /HeiseiMinchoUDC69Hi
    /HeiseiMinchoUDC70Lo
    /HeiseiMinchoUDC70Hi
    /HeiseiMinchoUDC71Lo
    /HeiseiMinchoUDC71Hi
    /HeiseiMinchoUDC72Lo
    /HeiseiMinchoUDC72Hi

    (sections 73Lo through 87Hi omitted)

    /HeiseiMinchoUDC88Lo
    /HeiseiMinchoUDC88Hi
    /HeiseiMinchoUDC89Lo
    /HeiseiMinchoUDC89Hi
```

     to:

```
%ADOStartRearrangedFont
/JHMNUDCH
  [/CS-K2093-H
    /HeiseiMinchoUDC69Lo
    /HeiseiMinchoUDC69Hi
%   /HeiseiMinchoUDC70Lo
%   /HeiseiMinchoUDC70Hi
%   /HeiseiMinchoUDC71Lo
%   /HeiseiMinchoUDC71Hi
%   /HeiseiMinchoUDC72Lo
%   /HeiseiMinchoUDC72Hi

    (sections 73Lo through 87Hi omitted)

%   /HeiseiMinchoUDC88Lo
%   /HeiseiMinchoUDC88Hi
%   /HeiseiMinchoUDC89Lo
%   /HeiseiMinchoUDC89Hi
```

  2. The rearranged CMaps already specify all the mappings for UDCs in the sections reserved for UDCs, so you do not have to change the mappings.

See “Rearranged fonts” on page 13 for a discussion of rearranged font files.

If you want to create a rearranged CMap that refers to another CMap, modify the CMap name. This line:

```
[ /CS-K2093-H
```

specifies the basis for the rearranged font. If you want to change the basis from IBDHK300.CMP to IBDH0300.CMP, change the CMap name to:

```
[ /CS-2093-H
```

Checking the ATMDATA.DAT file

Check that the values in the ATMDATA.DAT file are correct. The ATMDATA.DAT file maps the font name in the rearranged file (the alias name) to the file names in your file system. The alias name is to the left of the > (greater than) sign and the fully qualified file name is to the right.

Use Type Transformer DB to check and edit the ATMDATA.DAT. Select **Options –> Edit ATMDATA.DAT...**.

Creating or modifying the Type Transformer batch file

If you input values from the panel of Type Transformer DB, skip this section.

Type Transformer supplies sample batch files called *d*:\DUVTT\*lang**xxxx_kfet*.DBT, where:

d is the drive.

lang
: is CHS (Simplified Chinese), CHT (Traditional Chinese), JAPAN, or KOREA.

xxxx
: is the typeface:

SFSG	Simplified Chinese Fang Song
SHEI	Simplified Chinese Hei
SKAI	Simplified Chinese Kai
SSNG	Simplified Chinese Song
TKAI	Traditional Chinese Kai
TSNG	Traditional Chinese Sung
JHKG	Japanese Heisei Kaku Gothic
JHMG	Japanese Heisei Maru Gothic
JHMN	Japanese Heisei Mincho
HKG2	Korean Gothic
HSM2	Korean Myengjo

k indicates the character set: _ (Simplified Chinese GB, Traditional Chinese, Japanese, or Korean KS), K (Simplified Chinese GB18030), or U (Korean Full Hangul).

f indicates the format: R (raster) or O (outline).

e indicates what kind of font the batch file creates: B (base), E (extension), or X (extended base).

t indicates the type: C (DBCS Core) or S (DBCS Simulation).

In the Type Transformer batch file:

1. In the raster extended base batch files (*xxxxk*RX*t*.DBT) only, change the values of the CFNAME keywords from *nnn*X to *nnn*0. This change allows the coded fonts to reference font objects that have already been created.
2. Specify the UDC code page on the CPNAME keyword.
3. Specify the rearranged font CMap file on the HCMAPFILE keyword.
4. If you are generating outline font character sets, set the OLNEXT keyword to YES.
5. For outline base and extended base fonts, set the CAPTURE keyword to YES.
6. If you are generating raster font character sets, set the SECTION keyword to the sections you are modifying.

7. If you don't want to override any existing font objects, change the FONTPATH value.
8. If you don't want to override existing coded fonts, change the CFNAME, CFNAMESB, or FONTPATH keywords.
9. If appropriate, change the BOXSIZE or POINTSIZE value.

Running the Type Transformer batch job

Refer to the Type Transformer help information for running the Type Transformer batch job.

Modifying the coded font (simulation fonts only)

If you are not creating outline font UDCs that are used with DBCS Simulation fonts, skip this section.

You need to modify the coded font in order to print outline font UDCs in the correct position with DBCS simulation fonts as follows:
1. Use the AFP Font Editor to convert the coded font to a flat file (see *IBM Infoprint Fonts: Introduction to Type Transformer and Utilities for Windows*).
2. Use the AFP Font Editor to edit the flat file.
3. Use the AFP Font Editor to reconvert the flat file to a coded font.

Coded font values of Japanese simulation fonts

Table 8. Japanese Gothic simulated by Heisei Kaku Gothic

Coded font	HCI	VCI	HBO	VBO
XZG16F	16	16	2	1
XZG20F	20	24	3	1
XZG24F	24	30	3	1
XZG32F	32	32	2	1
XZG36F	36	36	3	0
XZG40F	40	40	3	0
XZG48F	48	48	4	1
XZG64F	64	64	5	1
XZH12V	12	30	3	0
XZH16V	16	32	2	0
XZH18V	18	36	2	0
XZH20V	20	40	2	0
XZH24V	24	48	3	0
XZH32V	32	64	2	0

Notes:
1. XZG*bx*B and XZG*bx*X take the same value as XZG*bx*F.
2. XZH*bx*D, XZH*bx*J, XZH*bx*N, XZH*bx*O, and XZH*bx*U take the same value as XZH*bx*V.

Table 9. Japanese Heisei Kaku Gothic simulated by Heisei Kaku Gothic

Coded font	HCI	VCI	HBO	VBO
XZE24F	24	24	2	1
XZE26F	26	26	2	1
XZE32F	32	32	2	1
XZE36F	36	36	3	1
XZE3XF	32	32	2	1
XZE40F	40	40	4	1
XZE44F	44	44	4	1
XZE48F	48	48	4	2
XZE52F	52	52	6	2
XZE64F	64	64	4	2
XZF12V	12	24	3	0
XZF13C	13	26	3	0
XZF16V	16	32	3	1
XZF18V	18	36	3	0
XZF1XV	16	32	2	1
XZF20V	20	40	3	0
XZF22V	22	44	3	0
XZF24V	24	48	3	0
XZF26V	26	52	3	0
XZF32V	32	64	4	0

Notes:
1. XZE*bx*B takes the same value as XZE*bx*F.
2. XZF*bx*D, XZF*bx*J,XZF*bx*O, and XZF*bx*U take the same value as XZF*bx*V.

Table 10. Japanese Round Gothic simulated by Heisei Maru Gothic

Coded font	HCI	VCI	HBO	VBO
XZR36F	36	36	3	1
XZR40F	40	40	3	1
XZR48F	48	48	4	1
XZR64F	64	64	4	0
XZS18V	18	36	1	0
XZS20V	20	40	4	0
XZS24V	24	48	4	0
XZS32V	32	64	2	0

Notes:
1. XZR*bx*B and XZR*bx*X take the same value as XZR*bx*F.
2. XZS*bx*D, XZS*bx*J, XZS*bx*O, and XZS*bx*U take the same value as XZS*bx*V.

Table 11. Japanese Mincho simulated by Heisei Mincho

Coded font	HCI	VCI	HBO	VBO
XZM16F	16	16	2	1
XZM24F	24	24	2	1
XZM26F	26	26	3	1
XZM32F	32	32	2	1
XZM36F	36	36	3	2
XZM40F	40	40	4	2
XZM44F	44	44	4	2
XZM48F	48	48	5	2
XZM52F	52	52	5	2
XZM64F	64	64	4	2
XZZ24F	24	24	2	1
XZN12V	12	24	2	0
XZN13V	13	26	2	0
XZN16V	16	32	2	0
XZN18V	18	36	2	0
XZN20V	20	40	4	0
XZN22V	22	44	3	0
XZN24V	24	48	3	0
XZN26V	26	52	4	0
XZN32V	32	64	2	1
XZY12V	12	24	2	1

Notes:
1. XZN*bx*B and XZM*bx*X take the same value as XZM*bx*F.
2. XZZ*bx*B takes the same value as XZZ*bx*F.
3. XZN*bx*D, XZN*bx*J, XZN*bx*N, XZN*bx*O, and XZN*bx*Y take the same value as XZN*bx*V.
4. XZY*bx*D, XZY*bx*J, XZY*bx*N, XZY*bx*O, and XZY*bx*U take the same value as XZY*bx*V.

Table 12. Japanese Heisei Mincho simulated by Heisei Mincho

Coded font	HCI	VCI	HBO	VBO
XZK16F	16	16	2	1
XZK24F	24	24	2	1
XZK26F	26	26	3	1
XZK32F	32	32	2	1
XZK36F	36	36	4	2
XZK3XF	32	32	3	1
XZK40F	40	40	3	2
XZK44F	44	44	5	2
XZK48F	48	48	4	2
XZK52F	52	52	6	2
XZK64K	64	64	6	2
XZL12V	12	24	2	0
XZL13V	13	26	2	0
XZL16V	16	32	2	0
XZL18V	18	36	2	0
XZL1XV	16	32	2	0
XZL20V	20	40	2	0
XZL22V	22	44	3	0
XZL24V	24	48	4	0
XZL26V	26	52	3	0
XZL32V	32	64	4	0

Notes:

1. XZK*bx*B takes the same value as XZK*bx*F.
2. XZL*bx*D, XZL*bx*J, XZL*bx*N, XZL*bx*O, and XZL*bx*U take the same value as ZXL*bx*V.

Coded font values of Korean simulation fonts

Table 13. Korean Gothic simulated by Gothic

Coded font	HCI	VCI	HBO	VBO
XAG16K	16	16	2	1
XZG24K	24	30	3	1
XZH08K	8	16	1	0
XZH12K	12	30	3	0

Note: XZG*bx*L takes the same value as XZG*bx*K.

Table 14. Korean Mincho simulated by Myengjo

Coded font	HCI	VCI	HBO	VBO
XZM24K	24	24	3	1
XZM32K	32	32	3	1
XZM36K	36	36	3	1
XZM40K	40	40	4	2
XZM48K	48	48	5	2
XZM64K	64	64	7	2
XZN12K	12	24	3	0
XZN16K	16	32	3	1
XZN18K	18	36	4	1
XZN20K	20	40	4	0
XZN24K	24	48	5	0
XZN32K	32	64	7	1
Note: XZM*bx*L takes the same value as XZM*bx*K.				

Coded font values of Simplified Chinese simulation fonts

Table 15. Simplified Chinese Gothic simulated by Hei

Coded font	HCI	VCI	HBO	VBO
XZG16P	16	16	2	1

Table 16. Simplified Chinese Song simulated by Song

Coded font	HCI	VCI	HBO	VBO
XZS26P	26	26	3	1
XZS32P	32	32	4	1
XZS40P	40	40	5	1

Coded font values of Traditional Chinese simulation fonts

Table 17. Traditional Chinese Gothic simulated by Sung

Coded font	HCI	VCI	HBO	VBO
XZG16T	16	16	2	1

Table 18. Traditional Chinese Ming simulated by Sung

Coded font	HCI	VCI	HBO	VBO
XZM24T	24	24	2	1
XZM32T	32	32	4	1
XZM40T	40	40	5	1

Installing font objects in your font library

In order to print your new UDCs, you must install the following font objects in your font library:
- The modified code page (raster fonts only)
- The new character set
- Any coded fonts you created

Troubleshooting

If you can't print the UDCs you created, here are some possible causes and solutions:

- If you added your UDC to the wrong part of the section, it will appear at a location you don't expect. Check that you added it to the low or high part of the section, whichever you intended.
- The ATMDATA.DAT file may not be complete. Check that all your CMaps (base and rearranged) are listed in the ATMDATA.DAT file and that all alias names are included and spelled right.
- Your batch file may not ask for the UDC to be created. Check the keyword combinations of FONT240, FONTOLN and OLNEXT to make sure that you are asking for the right section. Remember that if you request both raster and outline fonts, only outline fonts are created.

  If you are creating raster fonts, make sure that you set the FCSCREATE keyword to YES.
- You may not have installed all the modified font objects in your font library. Check these objects:
  - The modified code page (raster fonts only)
  - The new character set
  - Any coded fonts you created

Chapter 4. Converting AFP raster UDCs to Type 1 outline UDCs

You can convert certain UDCs from AFP raster format to Type 1 outline format. To do this, you must perform the following steps:

1. Use the AFP2FON utility to convert the UDCs to Windows 3.1 UDC format.
2. Use the Windows End User Defined Character (EUDC) editor to convert the Windows 3.1 UDCs to TrueType format.
3. Use FontLab to convert the TrueType UDCs to Type 1 outline format.

Converting from AFP raster format to Windows 3.1 UDC format

The AFP2FON utility converts font sections that contain UDCs from AFP raster format to Windows 3.1 UDC format. It supports Japanese, Simplified Chinese, and Traditional Chinese UDCs, with the following limitations:

- Input AFP raster UDCs should be the output of FROM, PMF or FLSF. The AFP2FON utility does not officially support the fonts generated by Type Transformer as input.
- Because the EBCDIC UDC range is bigger than the Windows UDC range, some host UDCs are mapped to code points outside the Windows UDC range. To avoid confusion, AFP2FON does not process these UDCs.

Table 19 shows how AFP2FON maps code points.

Table 19. Code conversion table

Language	Host code range	PC code range	Number of characters
Japanese	X'6941'–X'72EA'	X'F040'–X'F9FC'	1,880
Simplified Chinese	X'7641'–X'78FE'	X'AAA1'–X'AFFE'	564
	X'7941'–X'7C9F'	X'F8A1'–X'FEFE'	658
	X'7CA0'–X'7FF8'	Not converted	672
Traditional Chinese (Big5)	X'C241'–X'C661'	X'FA40'–X'FEFE'	785
	X'C662'–X'D64'	X'8E40'–X'A0FE'	2,983
	X'D649'–X'E0EA'	X'8140'–X'8DFE'	2,041
	X'E0EB'–X'E2FD'	Not converted	395

To use the AFP2FON utility, follow these steps:

1. Prepare the following resources:
   - **Font resources.**

     The coded font, character set, and code page for any font section that you want to convert must be in the same directory. If any of these files are not present, AFP2FON returns an error and quits the process. For example, if you want to convert sections 69 and 71 of X0G24F, the following resources must be in your font resource directory:

X0G24F	Coded font
C0G24F69	Character set for section 69
C0G24F71	Character set for section 71
T1G24F69	Code page for section 69
T1G24F71	Code page for section 71

If you do *not* want to convert a specific section, move the character set and code page for that section to another directory. AFP2FON skips this section and tries to continue conversion

- **Disk space.**

  During the conversion process, AFP2FON generates work files in your output directory. Make sure that the output directory has enough disk space for both the work files and the converted font sections.

2. Start AFP2FON.
3. Select **Language**.
4. Specify the coded font that you want to convert. Click the **Browse** button and select the drive, directory, and coded font name. Click **OK**.
5. Specify the file path for the output font. Click the **Browse** button and select the drive, directory, and output font name (extension .FON). Click **OK**.
6. Click **Convert** to start conversion.
   - Messages from AFP2FON utility are displayed in the **AFP2FON process message** field.
   - A completion message pops up when the conversion is complete.
   - If the conversion terminates because of an error, an error message pops up. Detect the error, referring to the messages in the **AFP2FON process message** field, and try again. One possible error is an invalid input font.
7. To save the messages into a file, click the **File Msg** button. Specify a drive, directory, and log file name (extension .MSG). Click **Save**.

Converting from Windows 3.1 format to TrueType format

You must now use the Windows EUDC editor to convert the .FON file from Windows 3.1 UDC format to TrueType format. The process varies according to the version of Windows you are using.

On Windows 95, Windows 98, or Windows NT®

For more information about this process, refer to the help information of the Windows EUDC Editor.

1. To start the Windows EUDC Editor, select **Start –> Program –>Accessories –> EUDC Editor**.
2. To specify the font that you will use the UDCs with, select **File –> Select Font**.
   - To use the UDCs with all fonts, select the **Standard EUDC** check box.
   - To use the UDCs with a specific font, check the **UDC considered typeface** check box, select **Connected font**, and specify the font name.

   Click **OK**.
3. To import the .FON file into the EUDC, select **File –> Import**. Select the .FON file and click **OK**.
4. Save the .FON file as a TrueType font. A file with the extension .TTE is created in your system font directory. On Windows NT, if you selected the **Standard EUDC** check box, the True Type font file is called C:\WINNT40\FONTS\EUDC.TTE.

5. To exit the Windows EUDC Editor, select **File –> Exit**.

On Windows 2000

For more information about this process, refer to the help information of the Windows EUDC Editor.

1. To start the Windows EUDC Editor, select **Start –> Program –>Accessories –> EUDC Editor**.
2. To specify the font that you will use the UDCs with, select **File –> Link Font**.
   - To use the UDCs with all fonts, select the **Link to all fonts** check box.
   - To use the UDCs with a specific font, check the **Link to a specified font** check box and specify the font name.

   Click **OK**.
3. To import the .FON file into the EUDC, select **File –> Import Bitmap Font**. Select the .FON file and click **OK**.
4. Save the .FON file as a TrueType font. A file with the extension .TTE is created in your system font directory. If you selected the **Link to all fonts** check box, the True Type font file is called C:\WINNT\FONTS\EUDC.TTE.
5. To exit the Windows EUDC Editor, select **File –> Exit**.

Converting from TrueType format to Type 1 format

Finally, you must use FontLab to convert the .TTE file from TrueType format to Type 1 format. To do this, follow these steps:

1. Make a copy of the .TTE file with the extension .TTF.
2. Start FONTLAB.EXE.
3. Select **File –> Open**; then select the .TTF file.
4. To save the file in Type 1 format, select **File –> Save As**. In the **Select File Type** field, select **Type1 Binary**. Specify a file name with the extension .PFB. Click **Save**.

Now go to “Creating and modifying the UDC code page” on page 30 to complete the process for creating UDCs.

Notices

Chapter 2, “CMaps, CIDFonts, and CID-keyed fonts” is based on *Adobe CMap and CIDFont Files Specification*, Technical Specification #5014 (Mountain View, CA: Adobe Systems Incorporated, 16 October 1995).

Trademarks

Index

Readers' Comments — We'd Like to Hear from You

Infoprint Fonts
Creating User-defined Characters

Publication No. G544-5854-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
G544-5854-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
IBM Printing Systems Division
Department H7FE, Building 004M
Information Development
P.O. Box 1900
Boulder, CO USA 80301-9817

Fold and Tape **Please do not staple** Fold and Tape

G544-5854-00

Cut or Fold
Along Line

Program Number: 5648–E77

Printed in U.S.A.

G544-5854-00