POSTSCRIPT™
Software From Adobe

Glyph Bitmap Distribution Format (BDF) Specification

Adobe Developer Support

Version 2.2

22 March 1993

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120

Adobe Systems Europe B.V.
Office Centre
Jozef Israëlskade 48c
1072 SB Amsterdam, Netherlands
31-20-6767-661

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomon, Minato-ku
Tokyo 105, Japan
03-3437-8950

PN LPS5005

Contents

Glyph Bitmap Distribution Format (BDF) Specification

1 Introduction

This document describes the Adobe™ Glyph Bitmap Distribution Format (BDF), which is intended to be easily understood by both humans and computers. The format described in this document is subject to change without prior notification.

Note *In this version of this document, the term "character" has been replaced, where appropriate, by "glyph" — the correct term for the elements of a font according to current industry usage. Keywords such as* CHARS *and* START-CHAR *have been retained for compatibility reasons.*

2 Tape Format

The Adobe Systems glyph bitmaps are typically distributed on magnetic tape. Each tape is 1600 BPI, nine track, un-labeled, and contains two or more files. Each file is followed by an end-of-file (EOF) mark. The last file on the tape is followed by two EOF marks. Physical records contain 512 bytes. The last physical record in a file (preceding an EOF mark) can contain fewer than 512 bytes.

Each file is encoded in the printable characters (octal 40 through 176) of USASCII plus carriage return and line feed. Each file consists of a sequence of variable-length lines. Each line is terminated by a carriage-return (octal 015) and line-feed (octal 012). The first file on the tape is the Adobe Systems Copyright notice. The files which follow are font files. The format of font files is described in the following sections.

Note *Font tapes can also be obtained in UNIX® tar format. Be sure to specify tar format. No other tape formats are currently supported by Adobe Systems.*

3 File Format

BDF files are distributed in an ASCII encoded, human-readable form. The file begins with global information pertaining to the font as a whole, followed by the information and bitmaps for the individual glyphs. This data represents the font for a single size and orientation. Metrics for multiple writing directions may be included in a single file.

A BDF file has the general form described in the following sections. Each item is contained on a separate line of text in the file. Items on a line are separated by spaces. Lines may be of unlimited length.

Values specified in the file will be one of the following types: string, number, or integer. A value of the type number can be either a real number or an integer, and can be signed or unsigned. It may or may not contain a decimal point or a leading minus sign.

Note *This version lifts the restriction on line length. In this version, the new maximum length of a value of the type* string *is 65535 characters, and hence lines may now be at least this long. While it is possible to use a version 2.2 font file with a version 2.1 parser, it is also possible for it to be incompatible.*

3.1 Global Font Information

The following keywords describe global information which is valid for the entire font:

STARTFONT *number*

STARTFONT is followed by a version number indicating the exact file format used (for example, *2.1*).

COMMENT *string*

One or more lines beginning with the word COMMENT. These lines can be ignored by any program reading the file.

CONTENTVERSION *integer*

(Optional) The value of CONTENTVERSION is an integer which can be assigned by an installer program to keep track of the version of the included data. The value is intended to be valid only in a single environment, under the control of a single installer. The value of CONTENTVERSION should only reflect upgrades to the quality of the bitmap images, not to the glyph complement or encoding.

FONT *string*

FONT is followed by the font name, which should exactly match the PostScript™ language **FontName** in the corresponding outline font program.

SIZE *PointSize Xres Yres*

SIZE is followed by the point size of the glyphs and the *x* and *y* resolutions of the device for which the font is intended.

FONTBOUNDINGBOX *FBBx FBBy Xoff Yoff*

FONTBOUNDINGBOX is followed by the width in x and the height in y, and the x and y displacement of the lower left corner from origin 0 (for horizontal writing direction); all in integer pixel values. (See the examples in section 4.)

METRICSSET *integer*

(Optional) The integer value of METRICSSET may be 0, 1, or 2, which correspond to writing direction 0 only, 1 only, or both (respectively). If not present, METRICSSET 0 is implied. If METRICSSET is 1, DWIDTH and SWIDTH keywords are optional.

SWIDTH
DWIDTH
SWIDTH1
DWIDTH1
VVECTOR

These metrics keywords may be present at the global level to define a single value for the whole font. The values may be defined at this level, yet overridden for individual glyphs by including the same keyword and a value in the information for an individual glyph. For a composite font containing a large number of ideographic glyphs with identical metrics, defining those values at the global level can provide a significant savings in the size of the resulting file.

Note *Version 2.1 of this document only allowed the metrics keywords SWIDTH and DWIDTH, and only at the glyph level. If compatibility with 2.1 is an issue, metrics should not be specified as global values.*

These keywords all have the same meanings as specified in section 3.2, Individual Glyph Information.

STARTPROPERTIES *n*

The optional word STARTPROPERTIES may be followed by the number of properties (n) that follow. Within the properties list, there may be *n* lines consisting of a word for the property name followed by either an integer or string surrounded by ASCII double quotation marks (ASCII octal 042). Internal quotation characters are indicated (or "quoted") by using two quotation characters in a row.

ENDPROPERTIES

The word ENDPROPERTIES is used to delimit the end of the optional properties list in fonts files containing the word STARTPROPERTIES.

3.2 Individual Glyph Information

The beginning of the section containing information for individual glyphs is specified by the CHARS keyword (followed by the number of glyphs in the font). Individual glyphs are delimited by the STARTCHAR and ENDCHAR keywords; between these keywords are metrics information and the bitmap data.

The glyph metrics data includes both the scalable width (the device independent width of the corresponding outline font glyph) and the width of the screen font glyph, expressed in device pixels. The scalable width can be used by applications for accurately calculating line widths and compensating for round-off errors.

The glyph data section is introduced by the keyword:

CHARS *nglyphs*

CHARS is followed by *nglyphs*, the number of glyphs that follow. To make sure that the correct number of glyphs were actually read and processed, error checking is recommended at the end of the file.

Each of the glyphs is then represented by the following:

STARTCHAR *string*

The word STARTCHAR followed by a string containing the name for the glyph. In base fonts, this should correspond to the name in the PostScript language outline font's encoding vector. In a Composite font (Type 0), the value may be a numeric offset or glyph ID.

Note *In versions of this document prior to 2.2, this value was limited to a string of 14 characters.*

ENCODING *integer (integer)*

ENCODING is followed by a positive integer representing the Adobe Standard Encoding value. If the character is not in the Adobe Standard Encoding, ENCODING is followed by –1 and optionally by another integer specifying the glyph index for the non-standard encoding.

SWIDTH *swx0 swy0*

SWIDTH is followed by swx0 and swy0, the scalable width of the glyph in x and y for writing mode 0. The scalable widths are of type *Number* and are in units of 1/1000th of the size of the glyph and correspond to the widths found in AFM files (for outline fonts). If the size of the glyph is *p* points, the width information must be scaled by *p*/1000 to get the width of the glyph in printer's points. This width information should be regarded as a vector indicating the position of the next glyph's origin relative to the origin of this glyph. SWIDTH is mandatory for all writing mode 0 fonts.

To convert the scalable width to the width in device pixels, multiply SWIDTH times *p*/1000 times *r*/72, where *r* is the device resolution in pixels per inch. The result is a real number giving the ideal width in device pixels. The actual device width must be an integral number of device pixels and is given by the DWIDTH entry.

DWIDTH *dwx0 dwy0*

DWIDTH specifies the widths in x and y, dwx0 and dwy0, in device pixels. Like SWIDTH, this width information is a vector indicating the position of the next glyph's origin relative to the origin of this glyph. DWIDTH is mandatory for all writing mode 0 fonts.

SWIDTH1 *swx1 swy1*

SWIDTH1 is followed by the values for swx1 and swy1, the scalable width of the glyph in x and y, for writing mode 1 (vertical direction). The values are of type Number, and represent the widths in glyph space coordinates.

DWIDTH1 *dwx1 dwy1*

DWIDTH1 specifies the integer pixel width of the glyph in x and y. Like SWIDTH1, this width information is a vector indicating the position of the next glyph's origin relative to the origin of this glyph. DWIDTH1 is mandatory for all writing mode 1 fonts.

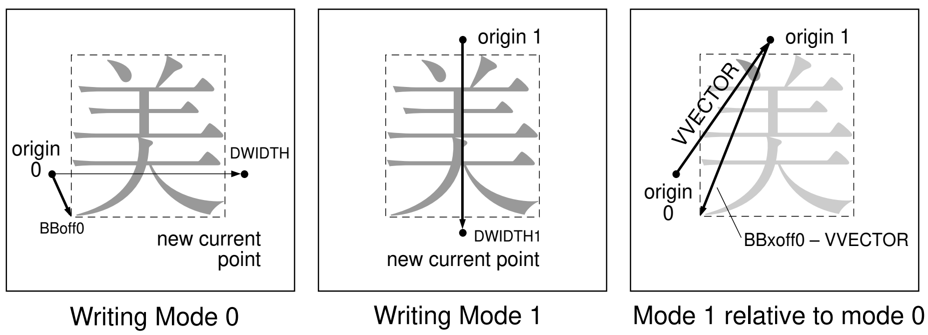
Note *If METRICSSET is 1 or 2, both SWIDTH1 and DWIDTH1 must be present; if METRICSSET is 0, both should be absent.*

VVECTOR *xoff yoff*

VVECTOR (optional) specifies the components of a vector from origin 0 (the origin for writing direction 0) to origin 1 (the origin for writing direction 1). If the value of METRICSSET is 1 or 2, VVECTOR must be specified either at the global level, or for each individual glyph. If specified at the global level, the VVECTOR is the same for all glyphs, though the inclusion of this keyword in an individual glyph has the effect of overriding the bal value for that specific glyph.

Figure 1 illustrates how VVECTOR relates the origins for writing directions 0 and 1.

Figure 1 *Relationship between metrics for writing direction 0 and 1*



BBX *BBw BBh BBxoff0x BByoff0y*

BBX is followed by BBw, the width of the black pixels in x, and BBh, the height in y. These are followed by the x and y displacement, BBxoff0 and BByoff0, of the lower left corner of the bitmap from origin 0. All values are are an integer number of pixels.

If the font specifies metrics for writing direction 1, VVECTOR specifies the offset from origin 0 to origin 1. For example, for writing direction 1, the offset from origin 1 to the lower left corner of the bitmap would be:

BBxoff1x,y = BBxoff0x,y – VVECTOR

BITMAP *<hex data>*

BITMAP introduces the hexadecimal data for the character bitmap. From the BBX value for *h*, find *h* lines of hex-encoded bitmap, padded on the right with zero's to the nearest byte (that is, multiple of 8). Hex data can be turned into binary by taking two bytes at a time, each of which represents 4 bits of the 8-bit value. For example, the byte 01101101 is two hex digits: 6 (0110 in hex) and D (1101 in hex).

ENDCHAR

ENDCHAR delimits the end of the glyph description.

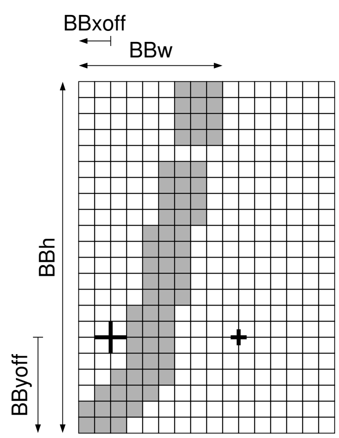
ENDFONT

The entire file is terminated with the word ENDFONT. If this is encountered before all of the glyphs have been read, it is an error condition.

4 Examples

Figures 2 through 4, along with Example 1, illustrate the bitmap format and glyph metric information. Figures 2 and 3 show examples of individual glyphs, which are both included in the font shown in Example 1. Figure 4 shows a glyph defined for use in both writing direction 0 and 1.

Figure 2 *Bitmap glyph and metrics for the glyph "j"*



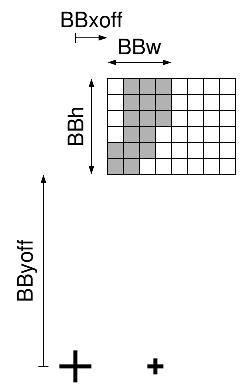

```
STARTCHAR j
ENCODING 106
SWIDTH 355 0
DWIDTH 8 0
BBX 9 22 -2 -6
```

In Figure 2, the bounding box is expressed differently than other PostScript language files such as the Adobe Font Metrics (AFM) file. The first two numbers following **BBX** are the width and height, the second two are the offsets in x and y.

The width from the origin (between the "+" indicators) is 8 pixels, which is how far the current point moves after rendering the character. It has nothing to do with the width of the bitmap.

The bounding box of the bitmap glyph can be used to predict how much data to read in the BITMAP section. The first two numbers give the width and height of the bitmap and correspond exactly to the amount of data supplied. The offset then allows positioning without repeating lots of white bits. (see the following *quoteright* glyph for an illustration of a glyph with all black pixels located a distance from the origin).

Figure 3 *Bitmap glyph and metrics for the "quoteright" glyph*



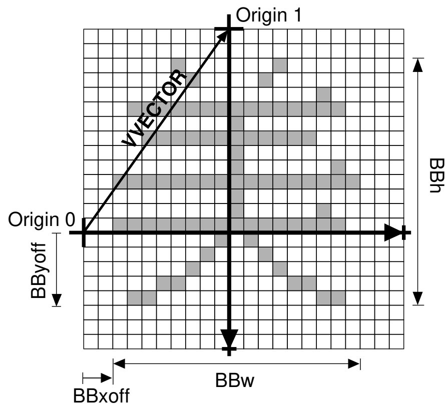

```
STARTCHAR
quoteright
ENCODING 39
SWIDTH 223 0
DWIDTH 5 0
BBX 4 6 2 12
```

In Figure 4, the actual bitmap is much smaller, and the offset (2 in x, 12 in y) positions the glyph with respect to its origin. The bitmaps in both Figure 2 and 3 are from an italic font program. Notice that the glyph width of the *quoteright* leaves the origin to the left of the black bits after the glyph is drawn, as would be expected for an italic font.

The bitmap is started by the BITMAP keyword and finished with the ENDCHAR keyword; this is illustrated in Example 1, below). It is best to predict the amount of data needed (using the BBX information) and use the ENDCHAR as an error-checking method. If you have consumed what you think is the appropriate amount of data, the next thing in the file should be ENDCHAR. If not, either the parser is in error, the file is not complete, or it is incorrect.

The bitmap is represented as hexadecimal digits, where each row corresponds to one row of the glyph bitmap. The bits are padded out to the nearest byte boundary with 0s. The BBX bounding box information should be carefully consulted to determine how to extract the data.

Figure 4 *Bitmap glyph and metrics for writing directions 0 and 1*




```
STARTCHAR c133
ENCODING 39
SWIDTH 1000 0
DWIDTH 22 0
BBX 17 17 2 -5
SWIDTH1 0 -1000
DWIDTH1 0 -22
VVECTOR 10 14
```

Example 1: *Sample BDF Font*

The following is an abbreviated example of a BDF bitmap file containing the specification of two glyphs (*j* and *quoteright* from Figures 2 and 3):

```
STARTFONT 2.1
COMMENT This is a sample font in 2.1 format.
FONT Helvetica-BoldOblique
SIZE 8 200 200
FONTBOUNDINGBOX 9 24 -2 -6
STARTPROPERTIES 2
MinSpace 4
Copyright "Copyright (c) 1987 Adobe Systems, Inc."
ENDPROPERTIES
CHARS 2
STARTCHAR j
ENCODING 106
SWIDTH 355 0
DWIDTH 8 0
BBX 9 22 -2 -6
BITMAP
0380
0380
0380
0380
0000
0700
0700
0700
0700
0E00
0E00
0E00
0E00
0E00
1C00
1C00
1C00
```

```
1C00
2C00
7800
F000
E000
ENDCHAR
STARTCHAR quoteright
ENCODING 39
SWIDTH 223 0
DWIDTH 5 0
BBX 4 6 2 12
BITMAP
70
70
60
E0
C0
ENDCHAR
ENDFONT
```

Appendix: Changes Since Earlier Versions

Changes since the January 16, 1989 version

- Document was reformatted in the new document layout and minor editorial changes were made.

Changes since the March 31, 1992 version

- The format was expanded to allow inclusion of metrics for the vertical writing direction, including new keywords METRICSSET, SWIDTH1, DWIDTH1, and VVECTOR.

- A new, global-level keyword CONTENTVERSION was added.

- The ability to specify metrics at a global level was added. If any of the metrics keywords occur before the first CHARS keyword, the values apply for all glyphs in the font. Global metric values can be overridden for individual glyphs by including the appropriate keyword and value at the individual glyph level.

- The restriction on string lengths (and hence line lengths) was lifted. Version 2.1 specified that the glyph name string following STARTCHAR was limited to 14 characters; version 2.2 allows the string length to be as long as the PostScript language string length of 65535 characters.

Index