ADOBE® FONT METRIC FILES Specification Version 3.0

March 8, 1990
PostScript® Developer Support Group

Adobe Systems Incorporated
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400

PN LPS5004

POSTSCRIPT®

ADOBE® FONT METRIC FILES Specification Version 3.0

March 8, 1990
PostScript® Developer Support Group
(415) 961-4111

1. INTRODUCTION

This document describes a standard interchange format for communicating font metric information to people and programs. The format is ASCII encoded (for both human and machine readability), machine independent, and extensible. Files in this format are known as Adobe® Font Metrics (AFM) and Adobe Composite Font Metrics (ACFM) files and are available for all of Adobe Systems' PostScript® language font software products.

1.1 DIFFERENCES FROM PRECEDING VERSIONS

This document supersedes the *Adobe Font Metric Files Specification, Version 2.0*. It extends the format of the AFM file to support composite font programs and multiple writing directions, and defines a new ACFM file to describe composite font programs.

The AFM format extensions are strictly upwards-compatible: a parser for AFM format version 3.0 can interpret version 2.0 metrics files correctly, and a parser for version 2.0 can extract that information from a version 3.0 metrics file which is compatible with the version 2.0 format. This is because parsers should ignore keywords they don't recognize.

1.2 BASE FONTS AND COMPOSITE FONTS

The PostScript language supports two kinds of font programs: *base fonts* and *composite fonts*. A base font is a font program as described by the *PostScript Language Reference Manual*, containing about 200 characters accessed by single-byte codes. A composite font program is a 'font of fonts'. Instead of containing characters, it contains *descendent font programs*, in a recursive, hierarchical structure. At the leaves of the structure are ordinary base font programs. Composite font programs can contain several thousand characters, accessed by multi-byte codes. They can be used for non-roman scripts, such as Japanese kanji.

Additionally, a font program may have several *writing directions*. The font program may contain several sets of metrics, and a key in the font dictionary selects which set of metrics is used to show a string. Both composite and base font programs allow several writing directions.

For more information, see the specification for the *PostScript Language Composite Font Extensions*.

1.3 AFM FILES AND ACFM FILES

This document describes two types of files: AFM (Adobe Font Metrics) and ACFM (Adobe Composite Font Metrics) files. An AFM file provides global metrics for a font program, and the metrics of each individual character. An ACFM file provides information about the structure of a composite font program — specifically, the global metrics of the composite font program, and the global metrics of each of its immediately descendent font programs. Their formats are sufficiently similar that a single parser can parse both AFM and ACFM files.

The metrics of a base font program are described completely by an AFM file. There is no ACFM file associated with a base font program.

The top-level structure of a composite font program is described by an ACFM file. The character metrics of individual characters in the composite font program are described by an associated AFM file[1].

The AFM file for a roman font program generally contains about 15 kilobytes. The AFM file for a Japanese kanji font program, using the JIS-83 character set, is about 200-360 kilobytes long. The ACFM file for that same JIS-83 kanji font program is about 35 kilobytes long.

1.4 STRUCTURE OF THIS DOCUMENT

Section 2 describes the syntax and data types of AFM and ACFM files, which are identical.

Section 3 describes the structure of an AFM file. Details of keywords and their parameters are deferred until sections 5 to 7.

Section 4 describes the structure of an ACFM file. Again, details of keywords and their parameters are largely deferred until section 5.

Section 5 defines the keywords and their parameters that apply to the global information about a font program. These keywords are used in both AFM and ACFM files.

Section 6 defines the keywords which apply to individual character metrics. These keywords are used only in the AFM file.

Section 7 defines the keywords which apply to kerning. These keywords are used only in the AFM file.

Section 8 defines the keywords which apply to composite characters. These keywords are used only in the AFM file.

Section 9 gives examples of AFM and ACFM files.

1. The ACFM file and corresponding AFM file for the top level of a composite font program generally contain all the information that most applications will need. Adobe ships only these files with its current composite font software products. However, because a composite font program has a hierarchical structure, it is meaningful to generate AFM files for its descendent font programs. Those descendent font programs can in turn be composite font programs, and in this case it is also meaningful to generate ACFM files for them.

2. PARSING DETAILS

Each AFM and ACFM file contains information for one PostScript language font program. Each file begins with global information pertaining to the font program as a whole, followed by, in an AFM file, sections with character metrics, or in an ACFM file, sections with descendent font program information. The file format is line-oriented, each line beginning with a property (key) name, followed by the values for that property. Keys and values are separated by one or more white space characters (space or tab).

The format is:

Key *value value* ...

Key names are case-sensitive. All keys beginning with a capital letter are reserved for use by Adobe Systems; user-defined non-standard entries must begin with a lowercase letter. The Adobe Systems standard keys are described below, but other keys are allowed and should be ignored by parsers not recognizing them.

Values will be one of the following types: **string**, **name**, **number**, **integer** or **boolean**. Each keyword unambiguously specifies the number and type of values that follow it.

- White space characters are ASCII space, newline, and tab.
- Empty lines, or lines containing white space characters only, may occur anywhere in the file and are ignored.
- Strings are terminated by the end of line. They begin with the first non-white-space character after the keyword, and may contain any printable ASCII character, plus space, tab, and ESC. Byte values greater than <7F> hex are not permitted.
- Names are similar to strings except that they may not contain any white space characters; they are terminated by white space characters.
- Numbers, integers and booleans are terminated by white space characters.
- A number can be either a real number or an integer, and signed or unsigned, i.e. it may or may not contain a decimal point or a leading minus sign.
- A boolean is either the value **true** or **false**.

With a few explicitly-noted exceptions, all keywords are optional. However, font software developers are strongly recommended to include as much information as is reasonable.

2.1 COMMENTS

Comments may be present in an AFM file. They are introduced by the keyword **Comment**, and are terminated by the end of line. Lines are no longer than 255 characters long.

Comment *string*

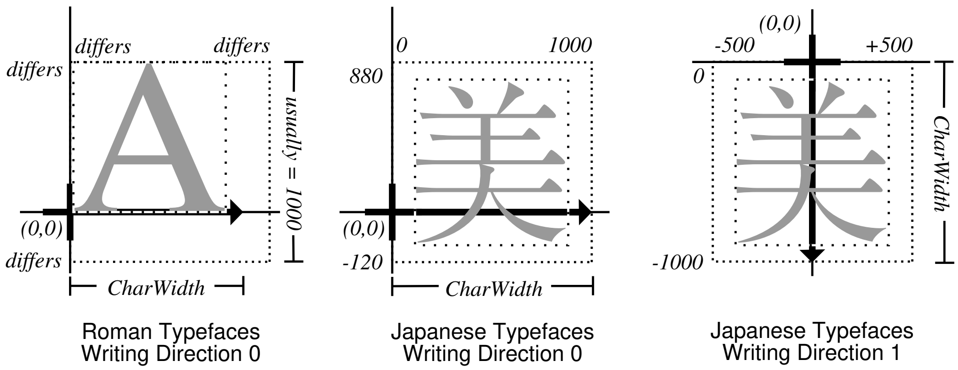
The text is arbitrary, and should be ignored.

2.2 UNITS OF MEASUREMENT

All measurements in an AFM and ACFM file are given in terms of units equal to 1/1000 of the scale factor (point size) of the font program. To compute actual sizes in a document, these amounts should be multiplied by (*scale factor of font*) / 1000.

The coordinate systems in which these units exist is defined by convention. For instance, the origin for Roman characters is on the baseline, a little to the left of the character, and the x-axis runs along the baseline. The figure below summarizes these conventions.

Figure 1: Character Coordinate Systems



Scale: 1000 = point size · · · · · · Character bounding box ········· Emsquare

3. AFM FILE STRUCTURE

An AFM file has several sections: a ‘header’ of control and global information for the font program, global writing direction information for the font program, and then several optional sections delimited by **Start**... and **End**... keywords.

StartFontMetrics *version*
EndFontMetrics

(Required.) These keywords delimit the entire AFM file. The **StartFontMetrics** keyword must be the first line in the file, and the **EndFontMetrics** keyword must be the last non-empty line in the file.

version is the version number of the AFM format specification used to generate this file. Fractional increases in the version number indicate minor, upwards-compatible revisions to the format; whole-number increases indicate major, potentially incompatible, changes.

3.1 CONTROL INFORMATION

The following key gives advance information about the structure of the data to follow.

MetricsSets *integer*

(Optional.) Which writing directions are described in this AFM file. *integer* may be 0, 1, or 2, meaning writing directions 0 only, 1 only, or both (respectively) are described. If not present, **MetricsSets 0** is implied. If present, it must occur in the header before the first **StartDirection** keyword.

3.2 GLOBAL FONT INFORMATION

The rest of the top-level section gives information which applies to all characters in the font program. In the case of a composite font program, it applies to all descendent font programs and to all characters in them. Only the keywords are listed below; they are defined and their parameters explained in section 5.1.

FontName *string*

FullName *string*

FamilyName *string*

Weight *string*

FontBBox *integer integer integer integer*

Version *string*

Notice *string*

EncodingScheme *string*

MappingScheme *integer*

EscChar *integer*

CharacterSet *string*

Characters *integer*

IsBaseFont *boolean*

VVector *number number*

IsFixedV *boolean*

CapHeight *number*

XHeight *number*

Ascender *number*

Descender *number*

3.3 WRITING DIRECTION INFORMATION

There are as many sections here as were specified by the top-level **MetricsSets** keyword. They are delimited by the following keywords:

StartDirection *integer*
EndDirection

which have the meaning defined in section 5.2. If omitted, **StartDirection 0** is implied. The following keywords may appear within **StartDirection** ... **EndDirection**. They also are defined in section 5.2.

UnderlinePosition *number*

UnderlineThickness *number*

ItalicAngle *number*

CharWidth *number number*

IsFixedPitch *boolean*

3.4 INDIVIDUAL CHARACTER METRICS

StartCharMetrics *integer*
EndCharMetrics

(Optional.) The section is delimited by the lines **StartCharMetrics** *integer* and **EndCharMetrics**. Each character's metrics consists of a list of keys and values separated by semicolons, on one line. A character metric data line might look like this:

C 102 ; **WX** 333 ; **N** f ; **B** 20 0 383 682 ; **L** i fi ; **L** l fl ;

These keywords and their parameters are defined in section 6.

3.5 KERNING DATA

StartKernData *integer*
EndKernData

(Optional.) The section is delimited by the lines **StartKernData** and **EndKernData**. Kerning data is supplied in two forms: track kerning and pair-wise kerning. They are treated as subsections within the kerning data section and both sections need not be present.

The kerning data keywords and their parameters are defined in section 7.

3.6 COMPOSITE CHARACTER DATA

StartComposites *integer*
EndComposites

(Optional.) If present, it is delimited by the lines **StartComposites** and **EndComposites**. Composite characters are new characters that are made up of characters already existing in the font program, such as accented characters.

The composite character keywords and their parameters are defined in section 8.

4. ACFM FILE STRUCTURE

ACFM files describe the *structure* of a composite font program. Base font programs do not have an ACFM file.

An ACFM file has three sections: a ‘header’ of control and global information for the top-level composite font program, writing direction information for the top-level composite font program, and then information for each of the immediately descendent font programs.

StartCompFontMetrics *version*
EndCompFontMetrics

(Required.) These keywords delimit the entire ACFM file. The **StartCompFontMetrics** keyword must be the first line in the file, and the **EndCompFontMetrics** keyword must be the last non-empty line in the file.

4.1 CONTROL INFORMATION

The following two keys give advance information about the structure of the data to follow.

Descendents *integer*

(Required.) Number of entries in the Encoding array of the PostScript language font program. (This is usually not equal to the number of distinct immediately descendent font programs.) There will be exactly this many entries in the descendent font program’s information section later in the ACFM files. This is keyword must occur before the first **StartDescendent** line.

MetricsSets *integer*

(Optional.) Which writing directions are described in this ACFM file. *integer* may be 0, 1, or 2, meaning writing directions 0 only, 1 only, or both (respectively) are described. If not present, **MetricsSets 0** is implied. If present, it must occur in the header before the first **StartDirection** keyword.

4.2 GLOBAL FONT INFORMATION

The rest of the top-level section gives information which applies to the topmost level of the composite font program, and also to all descendent font programs and all characters in them. Only the keywords are listed below; they are defined and their parameters explained in section 5.1.

This section of the ACFM file is identical to its counterpart in the same font program's AFM file.

FontName *string*

FullName *string*

FamilyName *string*

Weight *string*

FontBBox *integer integer integer integer*

Version *string*

Notice *string*

EncodingScheme *string*

MappingScheme *integer*

EscChar *integer*

CharacterSet *string*

Characters *integer*

IsBaseFont *boolean*

VVector *number number*

IsFixedV *boolean*

CapHeight *number*

XHeight *number*

Ascender *number*

Descender *number*

4.3 WRITING DIRECTION INFORMATION

There are as many sections here as were specified by the top-level **MetricsSets** keyword. They, too, are identical to their counterparts in the same font program's AFM file. They are delimited by the following keywords:

StartDirection *integer*
EndDirection

(Optional if **MetricsSets 0**, required otherwise.) These keywords have the same meaning as defined in section 5.2. If omitted, **StartDirection 0** is implied. The following keywords may appear within **StartDirection** ... **EndDirection**. They are also defined in section 5.2.

UnderlinePosition *number*

UnderlineThickness *number*

ItalicAngle *number*

CharWidth *number number*

IsFixedPitch *boolean*

4.4 DESCENDENT FONT INFORMATION

Following the global and writing direction information, there is information on the immediately descendent font programs. This descendent font program information follows the same format used in the 'header' portion of the AFM and ACFM files.

All descendent font programs must be described—the **Descendents** keyword has already indicated how many there are. The descendent font program information consists of a section for each descendent font program, delimited by the keywords:

StartDescendent *<ch-from> <ch-to>*
EndDescendent

(Required.) These comments introduce (and conclude) the information for one descendent font program. The two values are hexadecimal numbers, enclosed in angle brackets, which give the range of byte values which are mapped to this font program.
Example: <2100> <21FF>.

FontName *string*

FullName *string*

FamilyName *string*

Weight *string*

FontBBox *integer integer integer integer*

Version *string*

Notice *string*

EncodingScheme *string*

MappingScheme *integer*

EscChar *integer*

CharacterSet *string*

Characters *integer*

IsBaseFont *boolean*

VVector *number number*

IsFixedV *boolean*

CapHeight *number*

XHeight *number*

Ascender *number*

Descender *number*

Any of these keywords may be included if applicable to this descendent font program (and not already included at the top level). Their meanings are as described in section 5.1.

After the global information comes writing direction information for the descendent font program. There are as many sections here as were specified by the top-level **MetricsSets** keyword. They are delimited by the following keywords:

StartDirection *integer*
EndDirection

(Optional if **MetricsSets 0**, required otherwise.) These keywords have the same meaning and behavior as at the global level.

UnderlinePosition *number*

UnderlineThickness *number*

ItalicAngle *number*

CharWidth *number number*

IsFixedPitch *boolean*

Again, any of these keywords may be included if applicable to this descendent font program (and not already included at the top level). Their meanings are as described in section 5.2.

5. GLOBAL METRICS KEYWORDS

These keywords describe the global attributes of a font program in each of its writing directions. They appear at the top of AFM files and ACFM files, and within the descendent font program's information section of ACFM files.

5.1 GLOBAL FONT INFORMATION

The following global keys are the same as those in the top level or **FontInfo** subdictionary of the PostScript language font dictionary of an Adobe typeface. Their meanings are described in the chapter on font programs in the *PostScript Language Reference Manual*. Please note that although some of the keys in the **FontInfo** subdictionary begin with a lowercase letter (e.g., **isFixedPitch**, **version**), all keys listed here begin with uppercase letters to distinguish them as keys reserved for use by Adobe Systems. All numeric values are in units of 1 / 1000 of the scale factor of the font program.

FontName *string*

(Required.) Name of the font program as presented to the PostScript language **findfont** operator. *Examples*: Garamond-Light, Ryumin-Light-V.

FullName *string*

(Optional.) The full text name of the font.
Examples: ITC Garamond Light, Ryumin Light V.

FamilyName *string*

(Optional.) The name of the 'font family' to which the font belongs.
Example: ITC Garamond, Ryumin.

Weight *string*

(Optional.) Weight of the font. *Example*: Roman, Bold, Light.

FontBBox *integer integer integer integer*

(Required.) Four *integers* giving the lower left corner and the upper right corner of the font bounding box, in the sequence *llx lly urx ury*. *Note:* the bounding box given here is that of the flattened paths, not the Bézier curve descriptions. These values are all *integers*, and software generating them must round them outwards (i.e. *llx* and *lly* downwards, *urx* and *ury* upwards) if necessary.

Version *string*

(Optional.) Font program version identifier. Matches the string found in the **FontInfo** dictionary of the font program itself.

Notice *string*

(Optional.) Font name trademark or copyright notice.

EncodingScheme *string*

(Optional.) String indicating the default encoding vector for this font program. Common ones are *AdobeStandardEncoding* and *JIS12-88-CFEncoding*. Special font programs may simply state *FontSpecific*.

MappingScheme *integer*
(Not present with base font programs.) Integer code describing the mapping scheme. For details on mapping schemes, refer to the specification *PostScript Language Composite Font Extensions*. *Examples*: 2 (i.e. 8/8 mapping), 3 (escape mapping).

EscChar *integer*
(Required if **MappingScheme 3**, not present otherwise). The byte value of the escape character used for this escape-mapped font program.

CharacterSet *string*
(Optional.) String describing the character set of this font program. *Example*: ISO Latin-1.

Characters *integer*
(Optional.) The number of characters defined in this font program.

IsBaseFont *boolean*
(Optional.) *boolean* is **true** if this font program is a base font and **false** otherwise. If not present, this is assumed to be a base font program. Always **false** in the top level of an ACFM file.

VVector *number number*
(Required when **MetricsSets 2**.) Components of a vector from origin 0 (the origin for writing direction 0) to origin 1 (the origin for writing direction 1). If present, then the VVectors for all characters are the same (and **IsFixedV** is **true**); otherwise they are not the same.

IsFixedV *boolean*
(Optional.) If *boolean* is **true**, this indicates that VVector is the same for every character in this font. If this keyword is present, its value must not conflict with **VVector**; if absent, its value is assumed to be **true** if **VVector** is present, and **false** if **VVector** is absent.

CapHeight *number*
(Optional.) Top of capital *H*, measured in character coordinate system. If this font program contains no capital *H*, this keyword may be missing or *number* may be 0.

XHeight *number*
(Optional.) Top of lower case *x*, measured in character coordinates. If this font program contains no lower case *x*, this keyword may be missing or *number* may be 0.

Ascender *number*
(Optional.) For roman font programs: top of lower case *d*. If this font program contains no lower case *d*, this keyword may be missing or *number* may be 0.

Descender *number*
(Optional.) For roman font programs: bottom of lower case *p*. If this font program contains no lower case *p*, this keyword may be missing or *number* may be 0.

5.2 WRITING DIRECTION METRICS

There are sections here for as many writing directions as were specified by **MetricsSets**. They are delimited by the following keywords:

StartDirection *integer*
EndDirection

(Optional if **MetricsSets 0**, required otherwise.) *integer* is 0, 1, or 2. Encloses the metrics for writing direction *integer*; **StartDirection 2** implies that the metrics for both writing directions are the same. If **MetricsSets 0** was specified or implied, **StartDirection** ... **EndDirection** may be omitted, and **StartDirection 0** is implied.

UnderlinePosition *number*

(Optional.) Distance from the baseline for centering underlining strokes. This may be interpreted as a *y*-displacement for writing direction 0, and an *x*-displacement for writing direction 1.

UnderlineThickness *number*

(Optional.) This is the stroke width for underlining, expressed in character coordinate space (proportional to the characters in the font program).

ItalicAngle *number*

(Optional.) Angle (in degrees counter-clockwise from the vertical) of the dominant vertical strokes of the font. For unslanted fonts, this angle will be zero. *Example*: –12.

CharWidth *number number*

(Optional.) The *x* and *y* components of the width vector of this font program's characters. If present, it means that all characters in this font program have the same **CharWidth** for this writing direction, and implies **IsFixedPitch** is **true**.

IsFixedPitch *boolean*

(Optional.) If *boolean* is **true**, this indicates that the font program is a fixed pitch (monospaced) font. A value of **false** indicates a proportionally spaced font. If this keyword is present, its value must not conflict with **CharWidth**; if absent, its value is assumed to be **true** if **CharWidth** is present, and **false** if **Charwidth** is absent.

6. INDIVIDUAL CHARACTER METRICS

Each character's metrics are represented as a list of keys and values separated by semicolons, contained on one line. The characters are sorted by numerically ascending character code. Unencoded characters follow the encoded characters and are identified by character codes of **–1**. Byte codes for which no character is defined are not listed.

Example: A character metric data line might look like this:

C 102 ; **WX** 333 ; **N** f ; **B** 20 0 383 682 ; **L** i fi ; **L** l fl ;

This section appears in the AFM file only. It is optional.

StartCharMetrics *integer*
EndCharMetrics

(Required if individual character metrics are present.) These keywords delimit the character metrics section of the file. The *integer* value indicates how many individual characters to expect.

C *integer*

Decimal value of default PostScript language character code (–1 if unencoded).

CH <*hex*>

Same as **C**, but the character code is given in hexadecimal. *Example*: <84AF>.

(Either C or CH is required.)

WX *number*

W0X *number*

(Optional.) Character width in *x* for writing direction 0. *y* is 0.

W1X *number*

(Optional.) Character width in *x* for writing direction 1. *y* is 0.

WY *number*

W0Y *number*

(Optional.) Character width in *y* for writing direction 0. *x* is 0.

W1Y *number*

(Optional.) Character width in *y* for writing direction 1. *x* is 0.

W $number_x$ $number_y$

W0 $number_x$ $number_y$

(Optional.) Character width vector (*x, y*) for writing direction 0.

W1 $number_x$ $number_y$

(Optional.) Character width vector (*x, y*) for writing direction 1.

VV *number*$_x$ *number*$_y$

(Optional.) Same meaning as **VVector** in the global font program information, but for a single character.

N *name*

(Optional.) PostScript language character name.

B *llx lly urx ury*

(Optional.) Character bounding box where *llx, lly, urx,* and *ury* are all *numbers*. If a character makes no marks on the page (e.g. the space character), this field reads **B 0 0 0 0**, and these values are not considered when computing the **FontBBox**.

L *successor ligature*

(Optional.) Ligature sequence where *successor* and *ligature* are both *names*. The current character may join with the character named *successor* to form the character named *ligature*. Note that characters may have more than one such entry. (*See example above.*)

7. KERNING DATA

The kerning data section appears only in the AFM file. It is optional; it may or may not be present for a given font program. The section is surrounded by the lines **StartKernData** and **EndKernData**. Kerning data is supplied in two forms: track kerning and pair-wise kerning. Track kerning is applied to all characters uniformly, whereas pair-wise kerning is applied to specific character pairs. Track kerning and pair-wise kerning may be used independently or together (i.e., it is possible to apply track kerning to a line of text and then to apply pair-wise kerning after). The two forms of kerning data are treated as subsections within the kerning data section, and may independently be present or absent.

StartKernData
EndKernData

(Required if track or pair-wise kerning are present.) These keywords delimit the kerning section of the file.

7.1 TRACK KERNING

Track kerning may be specified for those writing directions mentioned by **MetricsSets**. The track kerning data is surrounded by the keywords:

StartTrackKern *integer*
EndTrackKern

(Required if track kerning data are present.) *integer* indicates how many different sets of track kerning data are present.

Normally track kerning is provided in different *degrees* of tightness. Within a track (a degree of tightness), the amount to decrease (or possibly increase) the amount of space between characters increases (or possibly decreases) with the point size of the font (e.g., for tight track kerning, the amount to decrease the space between characters at 6 point might be 0.1 points and at 72 point it might be 3.78 points). These distances are measured along the width of the characters: parallel to the x-axis for horizontal writing directions, and parallel to the y-axis for vertical writing directions.

The data itself begins with the keyword **TrackKern** and is followed by the track kerning information:

TrackKern *degree min-ptsize min-kern max-ptsize max-kern*

The *degree* is an integer where increasingly negative degrees represent tighter track kerning and increasingly positive degrees represent looser track kerning. *min-ptsize*, *min-kern*, *max-ptsize* and *max-kern* are all *numbers*. Since the track kerning is a linear function, the minimum and maximum cut-off values (point sizes) are provided along with the amount to track kern by at the point size. The kerning amounts are given relative to the point size. From those 4 values, the track kerning function can be derived. The track kerning function is a linear function. The equation for the line can be determined from the data provided and, therefore, the track kerning values for any point size can be determined. The track kerning values for any point size below/above the minimum/maximum point size are constant (the minimum kerning amount/maximum kerning amount).

The track kerning function $k(p)$, i.e. the amount of displacement in points, in terms of the font's point size p, may be expressed mathematically as follows:

$$k(p) = k_0 \qquad \text{for } p < p_0$$

$$k(p) = \frac{(k_1 - k_0) * (p - p_0)}{(p_1 - p_0)} + k_0 \qquad \text{for } p_0 \le p \le p_1$$

$$k(p) = k_1 \qquad \text{for } p > p_1$$

where p_0, k_0, p_1, k_1 are parameters to the keyword, **TrackKern** ***degree*** $p_0\ k_0\ p_1\ k_1$

See the last section of this document for an example of these keywords in use.
Below is a sample of text printed using these track kerning values.

Figure 2: Track Kerning

6 pt

no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.

12 pt

no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.

18 pt

no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.

7.2 PAIR-WISE KERNING

The pair-wise kerning data is surrounded by the keywords:

StartKernPairs *integer*
EndKernPairs

StartKernPairs0 *integer*
EndKernPairs

StartKernPairs1 *integer*
EndKernPairs

(Required if pair-wise kerning data are present.) *integer* indicates how many pairs to expect. **StartKernPairs** and **StartKernPairs0** denote writing direction 0; **StartKernPairs1** denotes writing direction 1. An AFM file may contain pair-wise kerning data for more than one writing direction, each delimited by the **StartKernPairs0** or **StartKernPairs1** ... **EndKernPairs**.

There will be one kerning pair per line. Each line will begin with a keyword of the form **KP**, **KPH, KPX**, or **KPY**.

KP $name_1$ $name_2$ $number_x$ $number_y$

Name of the first character in the kerning pair followed by the name of the second character followed by the kerning vector specified as an (*x, y*) pair. The kerning vector is the amount by which to move the second character by relative to the first character to position it properly. The kerning vector is specified in the standard character coordinate system. As with all metrics, in order to use this vector it is necessary to scale it by (*current point size*) / 1000.

KPH <hex_1> <hex_2> $number_x$ $number_y$

Same as **KP**, but the byte strings needed to generate the characters are given, rather than the names. *hex1* and *hex2* are hexadecimal values, enclosed in angle brackets.

KPX $name_1$ $name_2$ $number_x$

Name of the first character in the kerning pair followed by the name of the second character followed by the kerning amount in the *x* direction (*y* is zero). The kerning amount is specified in the units of the character coordinate system.

KPY $name_1$ $name_2$ $number_y$

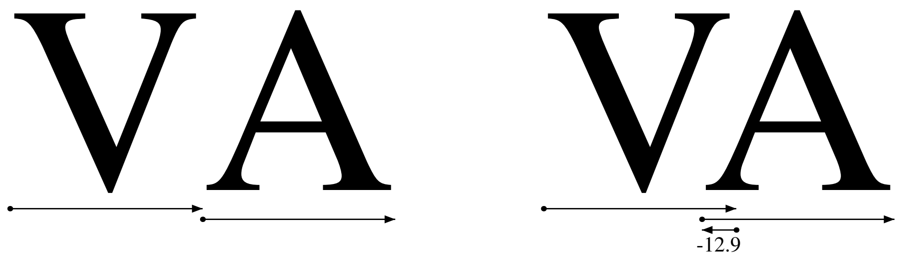
Same as **KPX**, but in the *y* direction (*x* is zero).

A character pair kerning line might look like this:

KPX V A -129

Below is an example of pair-wise kerning applied to 100 point characters:

Figure 3: Pair-wise kerning



Characters printed without kerning

Pair-wise kerning applied

8. COMPOSITE CHARACTER DATA

The composite character data section also appears only in AFM files. It is optional. Composite characters are new characters that are made up of characters already existing in the font program, such as accented characters. Character metric information for composite characters is found in the Character Metrics section of the AFM file. Although most PostScript language font programs available from Adobe Systems include a rather extensive set of composite characters, some applications may wish to generate their own. This section provides the data necessary for accurate positioning of the individual pieces. All units are expressed in the standard 1000 unit-per-font-scale-factor character coordinate system.

StartComposites *integer*
EndComposites

(Required if composite character data are present.) *integer* indicates how many pairs to expect.

The data for each composite character is represented as a list of keys and values separated by semicolons. Each composite character gets one line of description. The standard keys are:

CC *name integer*

The composite character name followed by the number of parts that make up the composite.

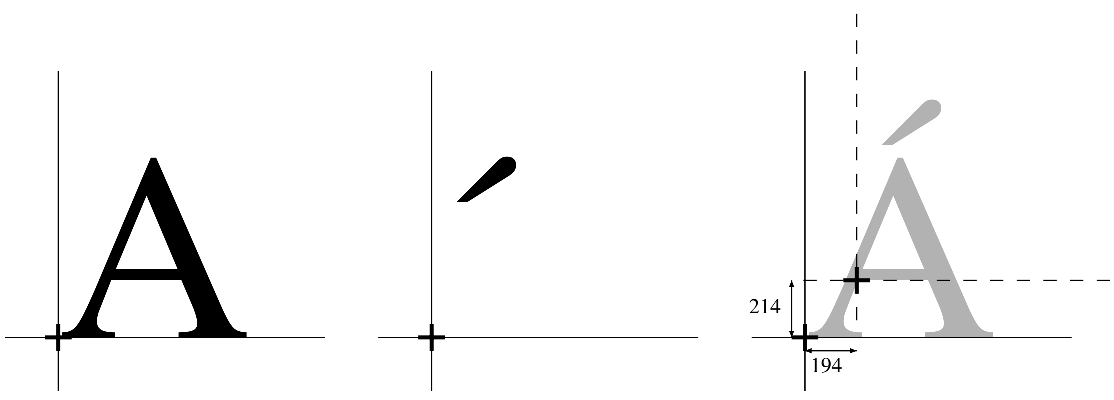
PCC *name deltax deltay*

One of the parts of the composite character. The character name is given followed by the *x* and *y* displacement from the origin.

A composite character line might look like this:

CC Aacute 2; **PCC** A 0 0; **PCC** acute 194 214;

Figure 4: Example of positioning for a composite character



Positioning of character A

Positioning of character acute

Composite Aacute

9. EXAMPLE AFM AND ACFM FILES

9.1 AFM FOR TIMES ROMAN

The following is an example of an AFM file for a roman base font program. All keywords which were not defined in version 2.0 of the AFM File Specification have been omitted, so that it is parseable both by programs which recognize the version 2.0 format and by those which recognize the version 3.0 format.

```
StartFontMetrics 3.0
Comment Copyright (c) 1985, 1987, 1989 Adobe Systems ...[truncated]
FontName Times-Roman
FullName Times Roman
FamilyName Times
Weight Roman
ItalicAngle 0
IsFixedPitch false
FontBBox -170 -223 1024 896
UnderlinePosition -109
UnderlineThickness 49
Version 001.004
Notice Copyright (c) 1985, 1987, 1989 Adobe Systems ...[truncated]
EncodingScheme AdobeStandardEncoding
CapHeight 662
XHeight 448
Ascender 682
Descender -217
StartCharMetrics 228
C 32 ; WX 250 ; N space ; B 0 0 0 0 ;
C 33 ; WX 333 ; N exclam ; B 109 -14 224 676 ;
C 34 ; WX 408 ; N quotedbl ; B 70 445 337 685 ;
C 35 ; WX 500 ; N numbersign ; B 4 0 495 662 ;
C 36 ; WX 500 ; N dollar ; B 44 -87 456 727 ;
C 37 ; WX 833 ; N percent ; B 61 -14 772 676 ;
 . . . -- lines omitted for brevity --
C 101 ; WX 444 ; N e ; B 22 -10 421 458 ;
C 102 ; WX 333 ; N f ; B 20 0 383 682 ; L i fi ; L l fl ;
C 103 ; WX 500 ; N g ; B 27 -217 470 458 ;
C 104 ; WX 500 ; N h ; B 9 0 490 682 ;
C 105 ; WX 278 ; N i ; B 22 0 259 682 ;
 . . . -- lines omitted for brevity --
C 249 ; WX 500 ; N oslash ; B 30 -108 470 549 ;
C 250 ; WX 722 ; N oe ; B 30 -10 690 458 ;
C 251 ; WX 500 ; N germandbls ; B 12 -10 468 682 ;
C -1 ; WX 611 ; N Zcaron ; B 7 0 597 888 ;
C -1 ; WX 444 ; N ccedilla ; B 25 -215 412 458 ;
C -1 ; WX 500 ; N ydieresis ; B 15 -217 476 623 ;
 . . . -- lines omitted for brevity --
 C -1 ; WX 750 ; N onehalf ; B 30 -14 720 676 ;
EndCharMetrics
StartKernData
StartTrackKern 3
Comment Light kerning
TrackKern -1 14 0 72 -1.89
Comment Medium kerning
TrackKern -2 8 0 72 -3.2
```

```
Comment Tight kerning
TrackKern -3 6 -.1 72 -3.78
EndTrackKern
StartKernPairs 113
KPX A y -92
KPX A w -92
 . . . -- lines omitted for brevity --
KPX y period -65
KPX y comma -65
EndKernPairs
EndKernData
StartComposites 58
CC Aacute 2 ; PCC A 0 0 ; PCC acute 195 214 ;
CC Acircumflex 2 ; PCC A 0 0 ; PCC circumflex 195 214 ;
 . . . -- lines omitted for brevity --
CC zcaron 2 ; PCC z 0 0 ; PCC caron 56 0 ;
EndComposites
EndFontMetrics
```

9.2 AFM FOR RYUMIN LIGHT VERTICAL

This is an example of the AFM file for a monospaced composite font program using writing direction 1. Since it is monospaced, the CharWidth keyword appears in the global **StartDirection** ... **EndDirection** section, and no pitch information is necessary in the individual character metrics section. The full file contains about 243 kilobytes.

The coordinate system for vertical Japanese font programs has its origin above the center of the character. The character is centered in a square ranging from −500 to +500 in *x* and from 0 to −1000 in *y*. Because vertical writing in Japanese goes from the top downwards, the **CharWidth** is negative in the *y* direction. The character <2121> is empty, and prints as a full-width blank.

```
StartFontMetrics 3.0
MetricsSets 1
FontName Ryumin-Light-V
IsBaseFont false
Characters 7238
FontBBox -500 -1185 500 23
EncodingScheme JIS12-88-CFEncoding
MappingScheme 2
FullName Ryumin Light 1983 JIS Standard Vertical ...[truncated]
Weight Light
Version 001.001
Notice Copyright (c) 1987, 1988, 1989 Adobe Systems  ...[truncated]
StartDirection 1
ItalicAngle 0
CharWidth 0 -1000
EndDirection
StartCharMetrics 7238
CH <2121> ; B 0 0 0 0 ;
CH <2122> ; B 211 -337 435 -81 ;
CH <2123> ; B 134 -370 430 -71 ;
CH <2124> ; B -406 -1020 -260 -760 ;
CH <2125> ; B -400 -900 -260 -760 ;
CH <2126> ; B -70 -592 71 -451 ;
CH <2127> ; B -71 -901 72 -214 ;
 . . . -- lines omitted for brevity --
CH <747D> ; B 0 0 0 0 ;
CH <747E> ; B 0 0 0 0 ;
EndCharMetrics
EndFontMetrics
```

9.3 AFM FOR RYUMIN LIGHT EXT HORIZONTAL

This is an example of an AFM for a composite font program with varying pitch. The characters in the font program are one of only two widths — full-width and half-width. This is an excerpt from the complete file, which contains about 315 kilobytes.

The coordinate system for horizontal Japanese font programs has its origin off the lower left side of the character. The character is centered in a square ranging from 0 to +1000 in *x* and from −120 to +880 in *y*. As before, the character <2121> is empty.

Note that no global **CharWidth** appears, and so pitch information is necessary in the individual character metrics section.

```
StartFontMetrics 3.0
MetricsSets 0
FontName Ryumin-Light-Ext-H
IsBaseFont false
Characters 7520
FontBBox -10 -305 1000 903
EncodingScheme ExtJIS12-88-CFEncoding
MappingScheme 2
FullName Ryumin Light Extended JIS Encoded Horizontal Composite Font
FamilyName Ryumin
Weight Light
Version 001.001
Notice Copyright (c) 1987, 1988, 1989 Adobe Systems ...[truncated]
StartDirection 0
ItalicAngle 0
EndDirection
StartCharMetrics 7520
CH <2121> ; W0X 1000 ; B 0 0 0 0 ;
CH <2122> ; W0X 1000 ; B 80 -75 329 170 ;
CH <2123> ; W0X 1000 ; B 80 -75 380 225 ;
CH <2124> ; W0X 1000 ; B 94 -140 240 120 ;
CH <2125> ; W0X 1000 ; B 100 -20 240 120 ;
CH <2126> ; W0X 1000 ; B 430 288 571 429 ;
CH <2127> ; W0X 1000 ; B 429 -21 572 666 ;
CH <2128> ; W0X 1000 ; B 423 -95 574 666 ;
 . . . -- lines omitted for brevity --
CH <2921> ; W0X 500 ; B 185 -40 315 788 ;
CH <2922> ; W0X 500 ; B 69 552 382 794 ;
CH <2923> ; W0X 500 ; B 87 0 413 725 ;
CH <2924> ; W0X 500 ; B 106 -128 400 831 ;
 . . . -- lines omitted for brevity --
CH <737D> ; W0X 1000 ; B 60 -63 967 827 ;
CH <737E> ; W0X 1000 ; B 54 -85 970 840 ;
EndCharMetrics
EndFontMetrics
```

9.4 ACFM FILE FOR RYUMIN LIGHT VERTICAL

This is an excerpt of the ACFM file for vertical Ryumin Light, of which the AFM file appeared in section 9.2. The information in the global font program section is identical to the corresponding information in the AFM file. However, the information for the descendent font programs, particularly the **FontBBox**, differs from descendent font program to descendent font program.

Note that there are 256 descendent font programs, corresponding to the 256 possible values for the first byte of a 2-byte character code, and the 256 entries in the PostScript language font program's **Encoding** array. Since the values of less than 21 hexadecimal or greater than 74 hexadecimal do not constitute legal JIS character codes, the corresponding descendent font programs are mapped to **NotDefFont**.

The full file contains about 34 kilobytes.

```
StartCompFontMetrics 3.0
MetricsSets 1
FontName Ryumin-Light-V
IsBaseFont false
Characters 7238
FontBBox -500 -1185 500 23
EncodingScheme JIS12-88-CFEncoding
MappingScheme 2
FullName Ryumin Light 1983 JIS Standard Vertical Composite Font
FamilyName Ryumin
Weight Light
Version 001.001
Notice Copyright (c) 1987, 1988, 1989 Adobe Systems ...[truncated]
StartDirection 1
ItalicAngle 0
CharWidth 0 -1000
EndDirection
Descendents 256
StartDescendent <0000> <00FF>
FontName NotDefFont
IsBaseFont true
Characters 0
EncodingScheme NotDefEncoding
EndDescendent
StartDescendent <0100> <01FF>
FontName NotDefFont
 . . . -- lines omitted for brevity --
EndDescendent
StartDescendent <2100> <21FF>
FontName Ryumin-Light.r21v
IsBaseFont true
Characters 94
FontBBox -495 -1020 495 0
EncodingScheme JISEncoding
EndDescendent
StartDescendent <2200> <22FF>
 . . . -- lines omitted for brevity --
EndDescendent
StartDescendent <7400> <74FF>
FontName Ryumin-Light.r74
```

```
IsBaseFont true
Characters 94
FontBBox -464 -964 481 -53
EncodingScheme JISEncoding
EndDescendent
StartDescendent <7500> <75FF>
 . . . -- lines omitted for brevity --
EndDescendent
StartDescendent <FF00> <FFFF>
FontName NotDefFont
IsBaseFont true
Characters 0
EncodingScheme NotDefEncoding
EndDescendent
EndCompFontMetrics
```

9.5 ACFM FILE FOR RKSJ-ENCODED KANJI FONT

This is the ACFM file for GothicBBB-Medium-83pv-RKSJ-H. This Japanese font program encodes roman characters and kana with single bytes, and kanji with Shift-JIS encoding. Its encoding is quite complex, but since there are so few descendent font programs the ACFM file is short, and is presented in its entirety. Some long lines have been truncated.

Note that the mapping ranges in the StartDescendent lines are either one byte or two byte, as appropriate. Since some characters are half width, CharWidth does not appear at the top level. The information in the descendent font program sections differs considerably from font program to font program.

There are 7 descendent font program entries, corresponding to the seven entries in the **Encoding** vector of this font program's PostScript language font dictionary. Let us suppose an application wants to make a copy of this font program and replace the 'UserGaiji' descendent font program with its own font program. Since the 'UserGaiji' font program is the sixth descendent font program, it is accessed via entry 5 of the **Encoding** array (entry 0 is the first entry); the application must modify this entry so that it refers to the new font program.

```
StartCompFontMetrics 3.0
MetricsSets 0
FontName GothicBBB-Medium-83pv-RKSJ-H
IsBaseFont false
Characters 8807
FontBBox -22 -252 1000 892
EncodingScheme FontSpecific
MappingScheme 6
FullName GothicBBB Medium JIS83 ShiftJIS Encoded ...[truncated]
FamilyName GothicBBB
Weight Medium
Version 001.001
Notice Copyright (c) 1987, 1988, 1989 Adobe Systems ...[truncated]
StartDirection 0
ItalicAngle 0
EndDirection
Descendents 7
StartDescendent <00> <7F>
FontName PCHelvetica
IsBaseFont true
Characters 95
FontBBox -22 -220 950 770
EncodingScheme FontSpecific
FullName PC Helvetica
FamilyName Helvetica
Weight Medium
Version 001.001
Notice Copyright (c) 1985, 1987 Adobe Systems ...[truncated]
StartDirection 0
ItalicAngle 0
UnderlinePosition -97
UnderlineThickness 73
EndDirection
EndDescendent
StartDescendent <80> <80>
FontName NotDefFont
```

```
IsBaseFont true
Characters 0
EncodingScheme NotDefEncoding
EndDescendent
StartDescendent <8100> <9FFF>
FontName GothicBBB-Medium-83pv-SuppA-H
IsBaseFont false
Characters 5828
FontBBox 0 -252 1000 892
EncodingScheme ExtShiftJIS-A-CFEncoding
MappingScheme 2
EndDescendent
StartDescendent <A0> <DF>
FontName GothicBBB-Medium.SuppK
IsBaseFont true
Characters 64
FontBBox 0 -114 500 874
EncodingScheme FontSpecific
StartDirection 0
CharWidth 500 0
EndDirection
EndDescendent
StartDescendent <E000> <EFFF>
FontName GothicBBB-Medium-83pv-SuppB-H
IsBaseFont false
Characters 2820
FontBBox 3 -252 1000 880
EncodingScheme FontSpecific
MappingScheme 2
StartDirection 0
CharWidth 1000 0
EndDirection
EndDescendent
StartDescendent <F000> <FBFF>
FontName UserGaiji
IsBaseFont false
Characters 0
EncodingScheme FontSpecific
MappingScheme 2
FullName User defined Gaiji Composite Font
FamilyName User Defined
Weight User Defined
Version 001.001
Notice Copyright (c) 1987, 1988, 1989 Adobe Systems ...[truncated]
StartDirection 0
ItalicAngle 0
EndDirection
EndDescendent
StartDescendent <FC> <FF>
FontName NotDefFont
IsBaseFont true
Characters 0
EncodingScheme NotDefEncoding
EndDescendent
EndCompFontMetrics
```

APPENDIX A: CHANGES SINCE EARLIER VERSIONS

Changes since January 16, 1989 version (Version 2.0)

- Major revision of AFM file format, from version 2.0 to version 3.0. Enhancements made to the format to reflect the *PostScript language Composite Font Extensions*.

- Section 1 (Introduction) greatly expanded. Differences from version 2.0 summarized. Global explanations of base and composite font programs, AFM and ACFM files added. Contents of each subsequent section summarized in advance.

- Section 2 (Parsing Details) slightly revised to make the syntax description more precise. Byte values above <7F> hex, and semicolons without spaces before them, are forbidden.

- New section 2.2 (Units of Measurement) added to explain the standard character coordinate systems for Roman and Kanji font programs.

- Section 3 (AFM File Structure, was simply File Structure) revised to apply to AFM file only. Section divided into sub-sections for each structural component of AFM file. Each sub-section refers to detailed keyword definitions in sections 5, 6, 7, 8.

- New keyword **MetricsSets** added to section 3.1 (Control Information). New keywords **MappingScheme**, **EscChar**, **CharacterSet**, **Characters**, **IsBaseFont**, **VVector**, **IsFixedV** added to section 3.2 (Global Font Information). Keywords **UnderlinePosition**, **UnderlineThickness**, **ItalicAngle**, **CharWidth**, **IsFixedPitch** moved to new direction-specific section 3.3 (Writing Direction Information), delimited by new **StartDirection** ... **EndDirection** keywords.

- New section 4 (ACFM File Structure) added. Corresponds to section 3, but describes ACFM file. New **StartCompFontMetrics** ... **EndCompFontMetrics** keywords delimit ACFM file. New keywords **MetricsSets** (same as in AFM file) and **Descendents** (not in AFM file) added to section 3.1 (Control Information). Sections 4.2 (Global Font Information) and 4.3 (Writing Direction Information) identical to sections 3.2 and 3.3. New section 4.4 (Descendent Font Information) added. New **StartDescendent** ... **EndDescendent** keywords delimit a set of keywords identical to those in sections 4.2 and 4.3.

- New section 5 (Global Metrics Keywords) added. Keywords referenced in sections 3.2, 3.3, 4.2, 4.3, and 4.4 defined here.

- Section 6 (Individual Character Metrics) was section 4 before. New **CH** keyword added to reflect multi-byte character codes. New **W0X**, **W1X**, **W0Y**, **W1Y**, **W0**, **W1**, and **VV** keywords added to reflect multiple writing directions.

- Section 7 (Kerning Data) was section 5 before. Section 7.1 (Track Kerning) essentially unchanged. New **StartKernPairs0** and **StartKernPairs1** delimiters, and new **KPH** and **KPY** keywords, added to section 7.2 (Pair-Wise Kerning).

- Section 8 (Composite Character Data) was section 6 before. Essentially unchanged.

- Section 9 (Example AFM and ACFM Files) was section 7 before. Times-Roman AFM file updated with current AFM. Example Ryumin-Light-V and Ryumin-Light-Ext-H AFM files, and Ryumin-Light-V and GothicBBB-83pv-RKSJ-H ACFM files, added.