Adobe® PostScript®

Adobe Font Metrics File Format Specification

Adobe Developer Support

Version 4.1

7 October 1998

Adobe Systems Incorporated

Adobe Developer Technologies
345 Park Avenue
San Jose, CA 95110
http://partners.adobe.com/

PN LPS5004

Contents

Adobe Font Metrics File Format Specification

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), Adobe Multiple Font Metrics (AMFM), and Adobe Composite Font Metrics (ACFM) files.

1.1 AFM, AMFM, and ACFM Files

This document describes four types of files: AFM, AMFM, ACFM, and CID-keyed AFM.

An AFM file provides both global metrics for a font program and the metrics of each individual character.

The metrics of a multiple master font program are described by one AMFM file, which specifies the control data and global font information, plus one AFM file for each of the master designs in the font program.

There are two font file formats used for Asian language Type 1 font programs with large character sets. The first, Original Composite Font (OCF) format fonts, have their metrics specified by an ACFM file and one or more associated AFM files.

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. The formats are sufficiently similar that a single parser can parse AFM, ACFM, and AMFM files.

The metrics of a base font program is 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.

Note *The ACFM file and corresponding AFM file for the top level of a composite font program generally contain all the information 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. The descendent font programs can in turn be composite font programs, and, in this case, it is also meaningful to generate ACFM files for them.*

The second format for large character set fonts is the CID-keyed (Character ID) font file format, where font metrics are specified by an AFM file with character metrics ordered by CID number. For a CID-keyed version of an AFM file, no ACFM file is required.

Note *CID-keyed fonts are, technically, composite fonts because more than one cross-referenced font file is used. However, in this document, the term composite font is reserved to describe OCF format fonts.*

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. The ACFM file for that same JIS-83 kanji font program is about 35 kilobytes long. The complete AFM files for an Japanese OCF font is about 2.7 megabytes, and for the corresponding CID-keyed AFM file it is about 400 kilobytes. For a multiple master font, the AMFM file is about 1.5 kilobytes, and the AFM for each master design is about 15 kilobytes.

1.2 Structure of This Document

Section 2 explains the four types of fonts described in this document. This includes base, composite, CID-keyed, and multiple master fonts.

Section 3 describes the syntax and data types of AFM, AMFM, and ACFM files.

Section 4 describes the structure of an AFM file. Details of keywords and their parameters are deferred until sections 5 to 7.

Section 5 describes the structure of an ACFM file. Again, details of keywords and their parameters are largely deferred until section 7.

Section 6 describes the structure of an AMFM file; details of the keywords and their parameters are described in section 7.

Section 7 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 8 defines the keywords which apply to individual character metrics. These keywords are used only in the AFM file.

Section 9 defines the keywords which apply to kerning. These keywords are used only in the AFM file.

Section 10 defines the keywords which apply to composite characters. These keywords are used only in the AFM file.

Section 11 gives examples of AFM, AMFM, ACFM and CID-keyed AFM files.

2 Font Types

The PostScript™ language supports three kinds of fonts: normal Type 1, multiple master (an extension of Type 1), and composite fonts (Type 0). The following sections describe the three types of font programs for which this document specifies the corresponding font metrics files. In addition, section 3.3 explains how to interpolate character metrics for a specific instance of a multiple master font program.

2.1 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 described in chapter 5 of the *PostScript Language Reference Manual, Second Edition*, 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 can support several writing directions. The font program can contain several sets of metrics, and a key in the font dictionary selects the set of metrics used to show a string. Both composite and base font programs allow several writing directions. For more information, see section 5.9 of the *PostScript Language Reference Manual, Second Edition*.

2.2 CID-Keyed Font Program Files

The CID-keyed font technology was designed to meet the performance and flexibility needs of large character set fonts, such as for Chinese, Japanese, and Korean language fonts. A CID-keyed AFM file does not utilize the ACFM and multiple AFM file structure; character metrics are listed only once, and are ordered by CID number. This results in significant savings in file size.

AFM files for Original Composite Font (OCF) files required one entry for each character for each encoding. This resulted in much duplication of information and huge files sizes.

Most Adobe OCF fonts have Composite style (ACFM and multiple AFMfiles) AFM files. There exist a few OCF fonts for which AFM files do not exist. CID-keyed AFM files are available for all Adobe Japanese fonts — both for OCF and CID-keyed fonts. CID-keyed fonts only have CID-keyed AFM files. All new Adobe Asian-language fonts will be issued in CID-keyed format and have accompanying CID-keyed AFM files.

For more information on the CID-keyed font file format, please see Adobe Technical Note #5014, “Adobe CMap and CIDFont Files Specification.” and Technical Note #5092, “Overview of the CID-Keyed Font Technology.” For a sample CID-keyed font AFM file, see section 11.

2.3 Multiple Master Font Programs

Multiple master font programs are an extension of the Type 1 font format, which allows the generation of a wide variety of typeface variations from a single multiple master font program. This capability allows users and applications unprecedented control over the typographic parameters of fonts used in their documents. This section gives a brief overview of multiple master font programs. For more details see “Adobe Type 1 Font Format: Multiple Master Extensions.”

A multiple master font program contains two or more typeface designs called master designs, which are organized to represent the dynamic range of one or more design axes, each representing a typographic variable such as weight, width, or optical size. This range of styles is defined in a multiple master font by specifying one master design to represent each end of an axis, such as a light and extra-bold weight, as well as any intermediate designs. The maximum number of master designs allowed is sixteen.

A font instance can be derived from a multiple master font program by specifying a weighted average of the master designs. The weighting values are specified by the value of the **WeightVector** array in the font dictionary. In a blend of *k* master designs, there are *k* weights that sum to 1.0 and determine the relative contributions of each master design to the resulting interpolated font. Each font instance can share the character descriptions (the **CharStrings** and **Private** dictionary) of the multiple master font from which it was derived, making it economical to generate a variety of styles.

The AMFM file for a multiple master font program contains global and control information for the font, as well as information about each design axis and master design. This file uses the typeface family name, for example, *MyriadMM*. The metrics for each master design are represented in a standard, separate AFM file. The name of each file is located in the information for each master design in the AMFM file (delimited by StartMaster and EndMaster keywords), for example, *MyriadMM-LightCn*.

Metric information such as a character's width for a particular font instance, can be derived by interpolation using the value from each master design and the **WeightVector** array from the font for which the metrics are to be calculated. The value of **WeightVector** can be obtained by a call to the Application Program Interface (API) of the Adobe Type Manager™ (ATM™) software, which is required for the use of multiple master typefaces. (See Technical Note #5074, "Adobe Type Manager Software API for Multiple Master Fonts: Macintosh.") Details of calculating interpolated metrics are discussed in section 3.3.

The AFM file serves as the source for metrics for all font instances derived from the multiple master font. In some environments such as Macintosh® or Windows™, the interpolated value or metrics for an instance of the font will be contained in the FOND Resource or in the *.PFM* file, in which case an application program need not do the interpolation.

A multiple master primary font is a font instance shipped with the font package and pre-calculated to match a standard style for the particular typeface family. It is backward-compatible with all existing systems and applications.

3 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 ignored by parsers not recognizing them.

Values will be one of the following types: **string**, **name**, **number**, **integer**, **array**, 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, can occur anywhere in the file after the first line and are ignored.
- Strings are terminated by the end of line. They begin with the first non-white-space character after the keyword, and can 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 might 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, that is, it may or may not contain a decimal point or a leading minus sign.
- Arrays can contain either names, numbers, integers, booleans, or other arrays.
- A boolean value is either **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.

3.1 Comments

Comments can 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*

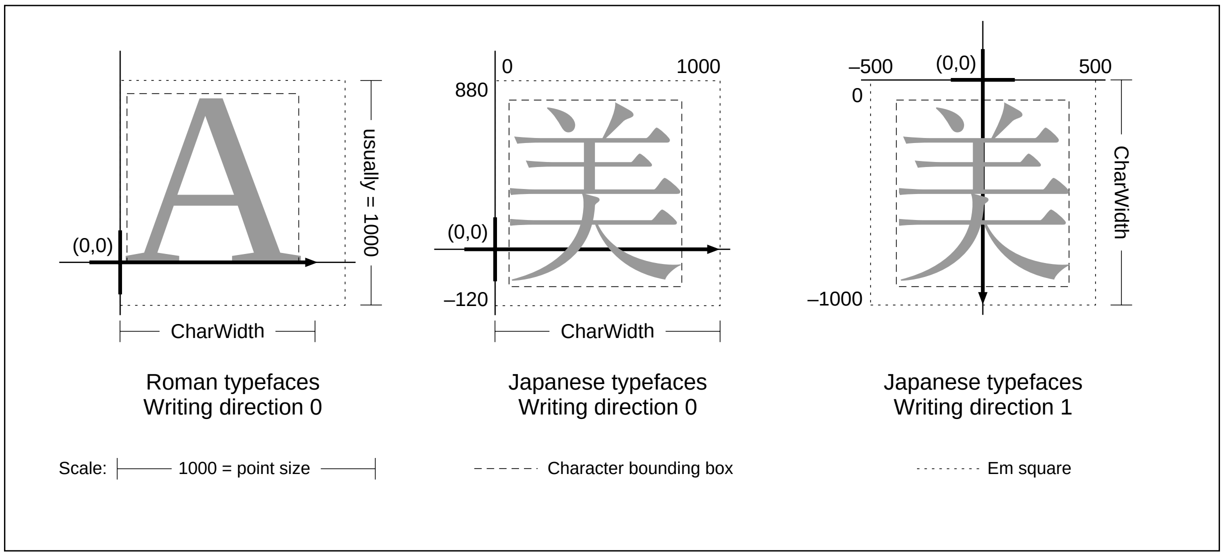
The text is arbitrary and should be ignored.

3.2 Units of Measurement

All measurements in AFM, AMFM, and ACFM files are given in terms of units equal to 1/1000 of the scale factor (point size) of the font being used. To compute actual sizes in a document (in points; with 72 points = 1 inch), 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. Figure 1 summarizes these conventions.

Figure 1 *Character coordinate systems*



3.3 Interpolating Metric Information for Multiple Master Font Programs

For multiple master fonts, applications that depend on AFM files must interpolate the metrics for the font instance chosen by the user. To do this, the application must calculate a weighted average of the widths for a character from each of the master designs in the font program. The widths for the master designs are taken from the AFM files that correspond to each master design.

If there are *k* master designs in a multiple master font, the font will have *k* values in **WeightVector** array that sum to 1.0. The interpolated value of a parameter such as a character width can be calculated by the following equation:

$$\text{Width} = \sum_{i=1}^{k} W_i \cdot V_i$$

where W_i is the width from the AFM file for the ith master design, and V_i is the ith element of the **WeightVector** array (corresponding to the ith master design). This blend operation applies to widths, side bearings, underline position and thickness, and all other numeric parameters that vary among the various master designs.

4 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 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.

4.1 Control Information

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

MetricsSets *integer*

(Optional.) The writing directions 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.

4.2 Global Font Information

The rest of the top-level section gives information that 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 7.1.

FontName *string*

FullName *string*

FamilyName *string*

Weight ***string***

FontBBox *number number number number*

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 are delimited by the following keywords.

StartDirection *integer*
EndDirection

which have the meaning defined in section 7.3. If omitted, StartDirection 0 is implied. The following keywords might appear within StartDirection ... EndDirection. They also are defined in section 7.3.

UnderlinePosition *number*

UnderlineThickness *number*

ItalicAngle *number*

CharWidth *number number*

IsFixedPitch *boolean*

4.4 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 8.

4.5 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 9.

4.6 Composite Character Data

StartComposites *integer*
EndComposites

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

The composite character keywords and their parameters are defined in section 10.

5 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 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.

5.1 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 keyword must occur before the first StartDescendent line.

MetricsSets *integer*

(Optional.) The writing directions 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.

5.2 Global Font Information

The rest of the top-level section gives information that 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 7.1.

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 *number number number number*

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*

5.3 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 7.2. If omitted, StartDirection 0 is implied. The following keywords might appear within StartDirection ... EndDirection. They are also defined in section 7.2.

UnderlinePosition *number*

UnderlineThickness *number*

ItalicAngle *number*

CharWidth *number number*

IsFixedPitch *boolean*

5.4 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 that are mapped to this font program.

Example: <2100> <21FF>.

FontName *string*

FullName *string*

FamilyName *string*

Weight *string*

FontBBox *number number number number*

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 can be included if applicable to this descendent font program (and not already included at the top level). Their meanings are described in section 7.1.

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 7.2.

6 AMFM File Structure

AMFM files describe the structure of a multiple master font programs. There are two sections: a header of control and global information, and a section for information for each of the master designs.

Multiple master font programs can have up to sixteen sets of typefaces included in a single font program. These master designs are organized to represent one or more design axes that represent the dynamic range of style variations from which all intermediate designs can be interpolated.

For example, a typical design axis would be one for the weight of the font, where the master designs might represent the ultra-light and ultra-bold endpoints of the axis. In addition to axis endpoints, master designs might be included to represent intermediate designs. The maximum number of master designs allowed is expressed by the equation $2^n + x = 16$, where n is the number of design axes, x is the number of intermediate designs, and 16 is the maximum number of designs.

6.1 The AMFM File

The AMFM file is delimited by the following keywords:

StartMasterFontMetrics version
EndMasterFontMetrics

(Required). The StartMasterFontMetrics keyword must be the first line in the file, and the EndMasterFontMetrics keyword must be the last non-empty line in the file.

6.2 Control Information

The following two keys give advance information about how many master designs are included in the font, and how many design axes they describe.

Masters *integer*

(Required.) Specifies the number of master designs in a multiple master font. Its value is the number of paired keywords StartMaster and EndMaster which follow. It must occur before the first StartMaster keyword and before the WeightVector keyword.

Axes *integer*

(Required.) Specifies the number of axes in a multiple master font. If present, its value is the number paired keywords StartAxis and EndAxis which must follow.

6.3 Global Font Information

The rest of the top-level (header) section specifies information that applies to the topmost level of the multiple master font and to the default version of the multiple master font specified by its WeightVector. Detailed semantics are described in section 7.1.

FontName *string*

FullName *string*

FamilyName *string*

Weight *string*

ItalicAngle *number*

IsFixedV *boolean*

UnderlinePosition *boolean*

UnderlineThickness *number*

FontBBox *number number number number*

Note *With the addition of multiple master font programs, the values for FontBBox have changed from type "integer" to "number."*

Version *string*

Notice *string*

EncodingScheme *string*

CapHeight *number*

XHeight *number*

Ascender *number*

Descender *number*

WeightVector *array*

(Required.) The values specified by WeightVector represent the relative weights for each master design in the multiple master font program. The global instance of this keyword applies to the default instance of the multiple master font program.

BlendDesignPositions *array*

(Required.) BlendDesignPositions is an array of *k* arrays giving the locations of the *k* master designs in the design space. Each location sub-array has *n* numbers giving the location of the design in the *n* dimensions of the design space, with a minimum value of zero and a maximum value of one.

BlendDesignMap *array*

(Required.) specifies the mapping of design coordinates to normalized blend space coordinates for the master designs on the current axis.

BlendAxisTypes *array*

(Required.) specifies the type of the current axis. The following is an example:

```
BlendAxisType [ /Weight /Width /OpticalSize ]
```

6.4 Axis Information

There will be axis information for as many axes as are specified by the Axes keyword. These sections are delimited by the keywords

StartAxis
EndAxis

(Required.) The axis information sections contain character strings which can be used in the user interface for creating and managing multiple master fonts.

AxisType *string*

(Optional.) The AxisType value specifies the kind of design axis represented by the current axis. Only AxisType values reserved by Adobe Systems can be used. The current reserved types are *Weight*, *Width*, and *OpticalSize*.

AxisLabel *string*

(Optional.) AxisLabel can have any value. It can be identical to the AxisType, or varied as needed. It is primarily used in user interfaces for the creation or use of multiple master font programs.

6.5 Master Design Information

An AFM file for a multiple master font contains global information for all parameters that do not vary for each master design. For example, the Version number and Notice would appear in the usual place. All information specific to each master design is enclosed between the following keywords:

StartMaster
EndMaster

(Required.) These keywords introduce and conclude the information for each master font.

Each StartMaster keyword will be followed by information that is global for that particular master, such as FontName, FullName, FamilyName, Version, and WeightVector. The value of the FontName can be used to find the name of the AFM file which contains the individual character metrics for the current master design.

FontName *string*

FullName *string*

FamilyName *string*

Version *string*

WeightVector *array*

(Required.) The value of the WeightVector array for a master design font will have a series of 0’s and one value of 1. The 1 indicates the position in the WeightVector in which the current master design is located. More details are explained in section 7.1.

6.6 Primary Font Information

An AFM file for a multiple master font ....

The primary font information provides information about any primary fonts associated with this multiple master font. Each primary font is represented as a list of keys and values separated by semicolons, contained on one line.

Example: A primary font entry might look like this:

```
PC 215 300 ; PL (LT) (CN) ; PN (Light Condensed) ;
```

StartPrimaryFonts *integer*
EndPrimaryFonts

(Optional.) These keywords delimit the primary fonts section of the file. The integer value indicates how many entries to expect.

PC *integer ...*

(Required.) The primary coordinate values. There should be as many integers as specified by the Axes keyword. This set of user design coordinates specifies the point in the design space for this primary font.

PL *string*

(Optional.) The primary labels is a list of strings giving the label for the associated coordinate for each axis in the design space. These labels can be used in conjunction with the primary coordinates to create a unique name for the primary font and should follow the conventions specified in the Adobe Technical Note #5088, "Font Naming Issues."

PN *string*

(Optional.) This specifies the more traditional name (??) for this primary font.

Example:

```
StartPrimaryFonts 4
PC 215 300 ; PL (LT) (CN) ; PN (Light Condensed) ;
PC 215 600 ; PL (LT) (NO) ; PN (Light) ;
PC 215 700 ; PL (LT) (SE) ; PN (Light Semiextended) ;
PC 400 300 ; PL (RG) (CN) ; PN (Regular Condensed) ;
EndPrimaryFonts
```

7 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, AMFM, and ACFM files, and within the descendent font program's information section of ACFM files.

7.1 Global Font Information

The following global keys are the same as those in the top level or **FontInfo** sub-dictionary 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, Second Edition*.

Note that although some of the keys in the **FontInfo** sub-dictionary begin with a lowercase letter (for example, **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 (point size) of the font being formatted.

FontName *string*

(Required.) Name of the font program as presented to the PostScript language **findfont** operator.

Examples: ITC Garamond-Light, Ryumin-Light-V

For CID-keyed fonts, the value of **FontName** should be the same name used as the value of the **CIDFontName** in the CIDFont file (which is usually the typeface name, without the encoding, character set, or writing direction specification).

FullName *string*

(Optional.) The full text name of the font.

Examples: ITC Garamond Light, Ryumin Light V

FamilyName *string*

(Optional.) The name of the typeface family to which the font belongs.

Example: ITC Garamond, Ryumin

Weight *string*

(Optional.) Weight of the font.

Example: Roman, Bold, Light

FontBBox *number number number number*

(Required.) Four *numbers* 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 were specified to be integers in version 3.0 and before, but with the addition of multiple master fonts, the use of "numbers" is allowed.*

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 might 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 section 5.9 of the *PostScript Language Reference Manual, Second Edition*.

Examples: 2 (that is, 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 (glyph complement) of this font program.

Examples: AdobeStandardLatin; AdobeStandardCyrillic

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.

IsCIDFont *boolean*

(Required if AFM is for a CID-keyed font.) If the boolean is *true*, the font is a CID-keyed font, and the metrics are in CID number order. If CID number is omitted, there is no character for that code point.

CapHeight *number*

(Optional.) Usually the y-value of the top of the capital *H*. If this font program contains no capital *H*, this keyword might be missing or *number* might be 0.

XHeight *number*

(Optional.) Typically the y-value of the top of the lowercase *x*. If this font program contains no lowercase *x*, this keyword might be missing or *number* might be 0.

Ascender *number*

(Optional.) For roman font programs: usually the y-value of the top of the lowercase *d*. If this font program contains no lowercase *d*, this keyword might be missing or *number* might be 0.

Descender *number*

(Optional.) For roman font programs: typically the y-value of the bottom of the lowercase *p*. If this font program contains no lowercase *p*, this keyword might be missing or *number* might be 0.

StdHW *number*

(Optional). This number specifies the dominant width of horizontal stems (measured vertically in character space units).

StdVW *number*

(Optional). This number specifies the dominant with of vertical stems (measured horizontally in character space units).

BlendAxisTypes *array*

(Required.) The value of BlendAxisTypes is an array of strings that specify the name of each axis in the order in which the master designs are organized in the multiple master font program. There will be one string for each axis defined in the font program. The names are registered by Adobe Systems. An example of the three entries for a standard 3-axis design would be

```
BlendAxisType [ /Weight /Width /OpticalSize ]
```

BlendDesignPositions *array*

(Required.) The value of BlendDesignPositions is an array of *k* arrays that give the locations of the *k* master designs in the design space. Each location sub-array has *n* numbers giving the location of the design in the *n* dimensions of the design space, with a minimum value of zero and a maximum value of one.

For example, if there were five master fonts as follows:

design 1: standard (middle of the design space)
design 2: light condensed
design 3: light expanded
design 4: bold condensed
design 5: bold expanded

The BlendDesignPositions array would then look like this:

```
BlendDesignPositions [[.5 .5] [0 0] [0 1] [1 0] [1 1]]
```

BlendDesignMap *array*

(Required.) The value of BlendDesignMap is an array of *n* arrays where *n* is the number of design axes contained in the multiple master font program. Each of the *n* arrays contains *m* sub-arrays that specify the mapping of design coordinates into normalized coordinates for that design axis. The minimum value allowed for *m* is two, and the maximum is twelve.

Example:

```
BlendDesignMap [[[200 0] [900 1]] [[200 0] [800 1] [[6 0] [24 0.5]
[72 1]]]
```

In the above example, the design coordinates for a three-axis font are mapped to their normalized coordinates. The first two axes have a direct mapping of design coordinate end-points to their normalized equivalents. In this example, the third axis is for optical size and has a range of design coordinates from 6 to 72 (representing a *point size*). This range has been mapped to be piecewise linear, with the 0.5 normalized center point mapped to the design coordinate of 24, rather than to the value of 39 which would be the linear mid-point value.

WeightVector *array*

(Required.) The WeightVector array specifies the factors for deriving a weighted average of the master designs in a multiple master font program. In an interpolation of *k* master designs, there are *k* weights that sum to 1.0 and determine the relative contributions of each master design to the resulting interpolated font. The number of elements contained in the array is specified by the Masters keyword. The global instance of this keyword applies to the default instance of the multiple master font program.

7.2 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** can be omitted, and **StartDirection 0** is implied.

UnderlinePosition *number*

(Optional.) Distance from the baseline for centering underlining strokes. This can 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, and is generally proportional to the stroke widths of 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 non-italic 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.

7.3 Multiple Master Axis Information

There are sections for as many axes as are specified by the **Axes** keyword. These sections are delimited by the following keywords:

StartAxis
EndAxis

(Required.) The axis-specific information included in this section consists of strings to identify the axis type. This is intended primarily for use in the user interface.

AxisType *string*

(Optional.) The AxisType value specifies the kind of design axis represented by the current axis. Only AxisType values reserved by Adobe Systems can be used. The current reserved types are *Weight*, *Width*, and *OpticalSize*.

AxisLabel *string*

(Optional.) The AxisLabel can have any value: it might be identical to the AxisType or varied as needed. It is primarily used in user interfaces for the creation or use of multiple master font programs.

8 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. Characters not encoded 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 entries to expect.

C *integer*

Decimal value of default character code (–1 if not encoded).

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 (for example, 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 can have more than one such entry. (See example above.)

9 Kerning Data

The kerning data section appears only in the AFM file. It is optional and 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 can be used independently or together (that is, 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 can 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.

9.1 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 (for example, 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 four values, the track kerning function can be derived. The track kerning function is a linear function. The equation for the line is 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*), that is, the amount of displacement in points, in terms of the font's point size *p*, can be expressed mathematically as follows:

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

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

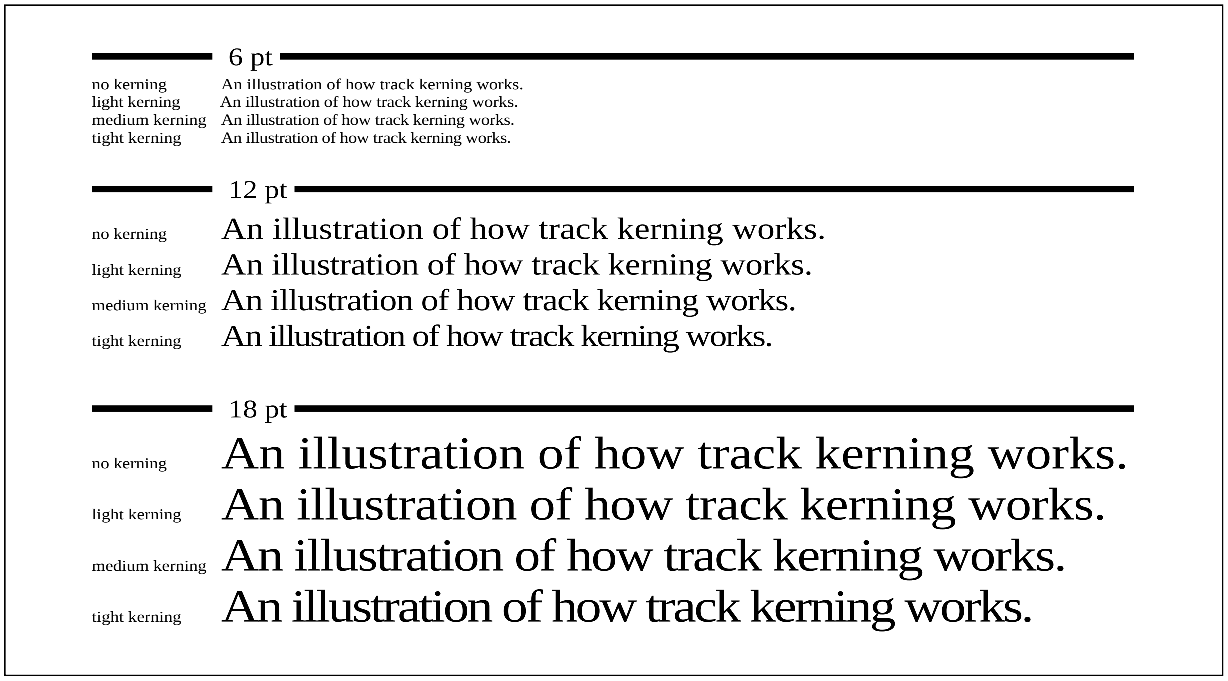
$$k(p) = k_1 \qquad \text{for } p > p_1$$

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.

9.2 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 the number of pairs to expect. StartKernPairs and StartKernPairs0 denote writing direction 0; StartKernPairs1 denotes writing direction 1. An AFM file can contain pair-wise kerning data for more than one writing direction, each delimited by the StartKernPairs0 or StartKernPairs1 ... EndKernPairs.

There is one kerning pair per line. Each line begins 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 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 $name_1$ $name_2$ $number_y$

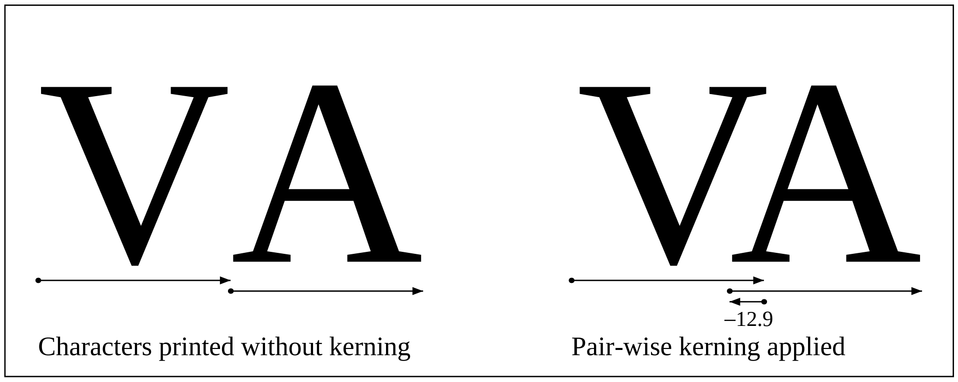
Same as KPX, but in the *y* direction (*x* is zero).

A character pair kerning line might look like this

```
KPX V A -129
```

Figure 3 is an example of pair-wise kerning applied to 100 point characters.

Figure 3 *Pair-wise kerning*



10 Composite Character Data

The composite character data section also appears only in AFM files. It is optional. Composite characters are new characters 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 might want 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 following are the standard keys:

CC *name integer*

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

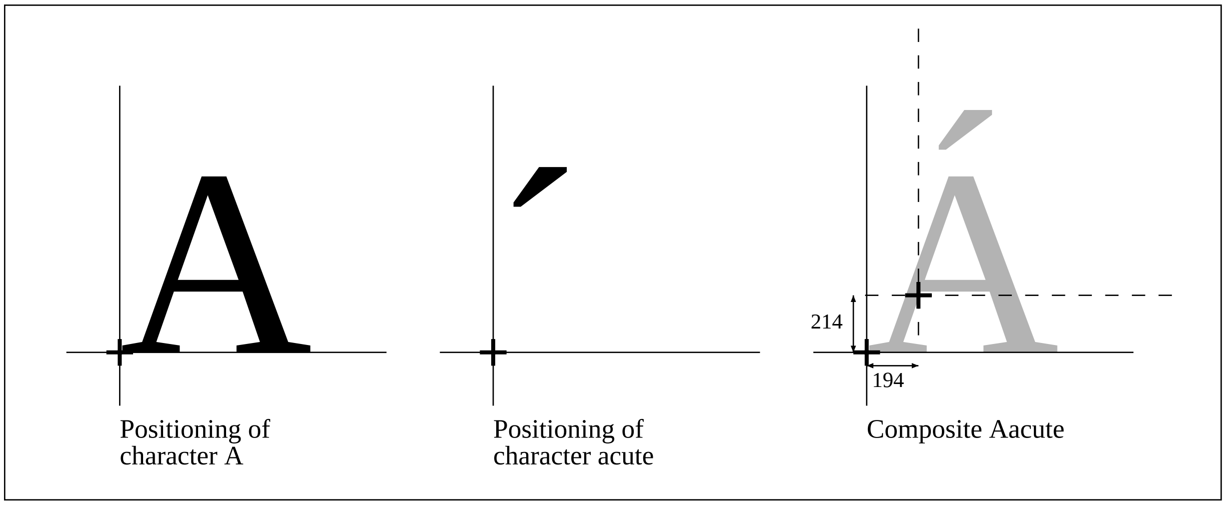
PCC *name deltax deltay*

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 *Positioning for a composite character*



11 Example AFM, AMFM, and ACFM Files

11.1 AFM File for Times Roman

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

```
StartFontMetrics 3.0
Comment Copyright 1985, 1987, 1989, 1990 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 1985, 1987, 1989, 1990 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
```

11.2 AMFM File for MyriadMM

The following is an example of an AMFM file for the Myriad™ multiple master font program.

```
StartMasterFontMetrics 4.0
Comment Copyright  1991, 1992 Adobe Systems Incorporated.  All Rights
Reserved.
Comment Creation Date: Tue Nov 27 14:35:40 1990
FontName MyriadMM
FullName Myriad MM
FamilyName  Myriad MM
Weight All
ItalicAngle 0
IsFixedPitch false
FontBBox -55.16 -250.00 1143.86 834.91
UnderlinePosition -100
UnderlineThickness 50
Version 000.009
Notice Copyright  1991, 1992 Adobe Systems Incorporated.  All Rights
Reserved.
EncodingScheme AdobeStandardEncoding
CapHeight 674
XHeight 483.72
Ascender 710.00
Descender -198.00
Masters 4
Axes 2
WeightVector [0.17 0.08 0.52 0.23 ]
BlendDesignPositions [[0 0] [0 1] [1 0] [1 1]]
BlendDesignMap [[[215 0][830 1]][[200 0][600 1]]]
BlendAxisTypes [/Weight /Width ]
StartAxis
AxisLabel Weight
AxisType Weight
EndAxis
StartAxis
AxisLabel Width
AxisType Width
EndAxis
StartMaster
FontName MyriadMM-LightCn
FullName Myriad MM Light Condensed
FamilyName Myriad MM
Version 001.000
WeightVector [1 0 0 0 ]
EndMaster
StartMaster
FontName MyriadMM-BlackCn
FullName Myriad MM Black Condensed
FamilyName Myriad MM
Version 000.009
WeightVector [0 1 0 0 ]
EndMaster
StartMaster
FontName MyriadMM-LightSemiEx
FullName Myriad MM Light Semi Extended
```

```
FamilyName Myriad MM
Version 000.009
WeightVector [0 0 1 0 ]
EndMaster
StartMaster
FontName MyriadMM-BlackSemiEx
FullName Myriad MM Black Semi Extended
FamilyName Myriad MM
Version 001.000
WeightVector [0 0 0 1 ]
EndMaster
EndMasterFontMetrics
```

11.3 AFM File 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 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
```

11.4 AFM File 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 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
```

11.5 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 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
```

11.6 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 seven descendent font program entries, corresponding to the seven entries in the Encoding vector of this font program's font dictionary. 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 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 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 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 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
```

11.7 AFM File for CID-Keyed Ryumin Light

This is an example of an AFM file for a CID-keyed font.

```
StartFontMetrics 4.0
MetricsSets 2
FontName Ryumin-Light
Weight Light
FontBBox -170 -331 1024 903
Version 3.003
CharacterSet Adobe-Japan1-2
Characters 8720
IsBaseFont true
IsCIDFont true
Ascender 903
Descender -331
CapHeight 903
StartDirection 2
UnderlinePosition -100
UnderlineThickness 50
ItalicAngle 0
IsFixedPitch false
EndDirection
StartCharMetrics 8720
C -1 ; W0X 1000 ; N 0 ; B 0 0 0 0 ;
C -1 ; W0X 250 ; N 1 ; B 0 0 0 0 ;
C -1 ; W0X 333 ; N 2 ; B 109 -14 224 676 ;
C -1 ; W0X 408 ; N 3 ; B 70 445 337 685 ;
C -1 ; W0X 500 ; N 4 ; B 4 0 495 662 ;
C -1 ; W0X 500 ; N 5 ; B 44 -87 456 727 ;
C -1 ; W0X 833 ; N 6 ; B 61 -14 772 676 ;
C -1 ; W0X 778 ; N 7 ; B 42 -14 750 676 ;
C -1 ; W0X 180 ; N 8 ; B 47 445 133 685 ;
C -1 ; W0X 333 ; N 9 ; B 49 -177 304 676 ;
. . . - - lines omitted for brevity - -
```

```
C -1 ; W0X 1000 ; N 8716 ; B 24 -89 936 848 ;
C -1 ; W0X 1000 ; N 8717 ; B 36 -68 955 832 ;
C -1 ; W0X 500 ; N 8718 ; B 24 194 474 475 ;
C -1 ; W0X 500 ; N 8719 ; B 42 -8 458 706 ;
EndCharMetrics
EndFontMetrics
```

Appendix: Changes Since Earlier Versions

Changes in Version 4.1, 7 October 1998

In section 7.1, "Global Font Information, the description of the /**FontName** keyword was changed to explain that for AFM files for CID-keyed fonts, the font name used should be the font name used as the value for the **CIDFontName** keyword in the CIDFont file.

Changes in Version 4.1, 25 September 1998

- Keywords have been added to AMFM files to describe the Primary Fonts that are created for, and ship with, multiple master fonts. The new key words include **StartPrimaryFonts**, **EndPrimaryFonts**, **PC**, **PL**, and **PN**.

- Two new keywords, **StdHW**and **StdVW** were added to specify the dominant weights of horizontal and vertical strokes.

- The format of an AFM file for CID-keyed fonts was introduced; the associated new keyword is **IsCIDFont**.

Changes in Version 4.0, 16 October 1995

- Revision in version 4.0 to describe the AMFM file format for use with Multiple Master font programs.

- A new section 2 includes description of font types, which includes a section on base fonts and ACFM's previously in section 1 of version 3.0. New section 2.2 gives an overview of multiple master font programs.

- Section 3.3 added to describe how to calculate a weighted average for values from multiple master font instances.

- A new section 6 describes the structure of AMFM files. A value type of *array* added for use by AMFM keywords. FontBBox value types changed from *integer* to *number.*

- Section 11 (which was section 9, “Example AFM and ACFM Files”) now includes a new subsection 11.2, “AMFM File for MyriadMM” as an example of a multiple master AMFM file.

Changes in Version 3.0

- 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 4, 5, 6, and 7.

- 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 4.1 (Control Information). Sections 5.2 (Global Font Information) and 5.3 (Writing Direction Information) identical to sections 4.2 and 4.3. New section 5.4 (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 4.2, 4.3, 5.2, 5.3, and 5.4 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.

Index