Supporting Fonts in the PostScript Language Environment

Adobe Developer Support

Technical Note #5075

31 March 1992

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support
Fax: (415) 961-3769

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120
Fax: (617) 273-2336

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O, Netherlands
+31-20-6511 200
Fax: +31-20-6511 300

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomon, Minato-ku
Tokyo 105, Japan
81-3-3437-8950
Fax: 81-3-3437-8968

PN LPS5075

Contents

List of Figures

Supporting Fonts in the PostScript Language Environment

1 Introduction

There are a number of technical and user interface issues that software applications must address to present users with a good solution for viewing and printing text. The advent of desktop publishing has brought an ever-increasing number of users who expect high-quality screen and printer fonts, a wide variety of typeface styles, and user-friendly font installation, selection, and printing.

This document gives an overview of how software applications can achieve an optimal font solution by supporting the PostScript™ language and the Adobe™ Type 1 font format. Although some platform-specific examples are discussed, this document mostly discusses general issues involved in how applications handle font support.

2 The PostScript Language Font Solution

Adobe's PostScript language is a powerful tool for describing the appearance of text and graphics on a display or printed page and has become the industry standard in this area. The PostScript language enables users to create device-independent documents that can be printed on a variety of devices ranging from dot matrix printers to high resolution imagesetters, as well as color printers, plotters, and engraving machines. PostScript language documents and fonts can also be transferred across platforms, giving users unprecedented flexibility for working in networked environments.

The two essential resources for PostScript language font support are the Adobe Type Manager™ and the Type 1 font programs from the Adobe Type Library. Sections 2.1 and 2.2 describe these resources and explain their value to application developers.

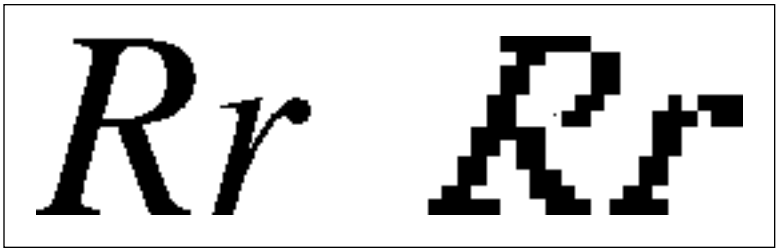
2.1 Adobe Type Manager Software for Display and Printing

The Adobe Type Manager (ATM™) software module converts outline format characters into a bitmap representation (a process called rasterization) for screen display or for printing to non-PostScript language output devices. It is currently available for the Macintosh®, OS/2®, and Windows™ 3.0 environments.

ATM software intercepts the standard text system calls made by an application, rasterizes the required characters from the outline font, and displays the resulting bitmap on the screen. This produces much more accurate bitmap characters than can be achieved by scaling a bitmapped screen font to the required size. Figure 1 shows the difference between a character generated by the ATM software and a scaled bitmap screen font character.

The advantage of generating screen fonts from the outline font used for printing is that the user gets WYSIWYG (what-you-see-is-what-you-get) capability. The term WYSIWYG describes applications that attempt to represent on the screen exactly what will be printed, so that line lengths, the number of lines on a page, and the position of text and graphics will print exactly as expected. This aids productivity because changes are more quickly reflected on the display than by printing. Users can then visualize changes and make decisions, thus shortening the development cycle.

Figure 1 *ATM software generated characters for a 72 dot-per-inch screen versus characters scaled from a bitmap screen font*



ATM software also offers software developers and users the ability to send output to less expensive, non-PostScript printers such as low-cost dot matrix printers, as well as to medium resolution devices such as the Hewlett-Packard® DeskJet® or the Apple® StyleWriter®. This gives even low-budget users the ability to use the wide variety of available Type 1 font programs and, in many cases, to achieve higher quality output than was previously possible with those printers.

One of the most significant benefits to users is that they can invest in typefaces for their current printer, and those fonts will work on a wide variety of output devices, as well as on any PostScript language device to which they

might upgrade in the future. In contrast, some brands of font rendering software for Windows, for example, require the purchase of proprietary format fonts that are compatible only with that software and platform.

Users appreciate applications that support a font format applicable to a wide variety of applications, printers, and platforms. This, in turn, benefits software developers by allowing their applications to appeal to a wider spectrum of users who will appreciate the flexibility of the supported font format.

The ATM software also offers developers an important resource by supporting an application program interface (API). Applications can call the ATM software to get standard or transformed (scaled, rotated, or skewed) character bitmaps or to get character outlines for further transformations or other special applications. Also, the ATM software can be called to draw and fill Bézier curves for screen display.

For more details, see Adobe Technical Note #5072, “Advanced Type Capabilities Using Adobe Type Manager Software on the Macintosh.”

2.2 The Type 1 Font Format and the Adobe Type Library

An outline format Type 1 font is called a font program—it is an executable piece of software which, when executed by a PostScript interpreter, defines a set of character and symbol outlines that can be referenced by PostScript language documents. Because the font descriptions are software programs, font developers can protect their typefaces as copyrightable software. (Copyright protection benefits both font vendors and end users by trying to prevent illegal copying and hence encouraging new typeface development.)

There are currently over 1300 fonts in the Adobe Type Library and over 13,000 Type 1 fonts available from all font vendors. There are Type 1 font programs for virtually every modern language and alphabet. Part of the Adobe Type Library is the Adobe Originals™ series, which features creative new designs and definitive revivals of classic styles. This series also offers extended character sets in the text fonts, which bring a much richer complement of characters to help users communicate better.

Type 1 format font programs offer many advantages to both users and software developers. A Type 1 font program describes character shapes with mathematically expressed curves and straight lines. The resulting outline font is much more versatile than a bitmap format font. One outline description of a typeface can be scaled to any point size, rotated to any angle, or used for a variety of other transformations (see Figure 3). The advantage is not only the accuracy and ease of doing this, but also that only one copy of a font need be stored on a disk, instead of one font for each size and orientation.

Type 1 font programs contain information, called *hints*, that aid the PostScript interpreter in rendering characters for all sizes and resolutions. The hinting method of the Type 1 format is relatively simple: Most hints are declarative statements stating where key features of a character are located. The intelligence for adjusting those features to look correct at any resolution, and to account for the artifacts of raster devices, exists in the interpreter. Consequently, the resulting fonts are of minimal size and are capable of improvement as interpreter algorithms improve.

Most typefaces in any type library are proportionally spaced type designs. Compared to monospaced faces, proportional space fonts are more legible, have a much broader variety of styles, and save a significant amount of space (see Figure 2). However, they require the application to do more work to access the metrics file and keep track of line widths.

Figure 2 *Monospaced Courier and proportionally spaced Times* Roman*

In addition to the Type 1 downloadable font programs sold by Adobe Systems, fonts can exist in several forms and locations within a user's system. Printer manufacturers usually bundle a core set of fonts with their printer; they reside in the printer's ROM, in a cartridge, or on an internal hard disk. The advantage of fonts residing in the output device is that they do not need to be downloaded from the host computer to the printer, which can save a significant amount of time. Cartridge and Disk fonts are classified as Type 4 and 5 fonts, respectively, but are similar to the Type 1 font format except for file organization.

The Adobe composite font language extensions provide for a hierarchical structure of fonts that removes the Type 1 font restriction that only 255 characters can be encoded. This capability is important for Asian ideographic scripts such as Kanji, and can be useful for latin font vendors who wish to provide larger character sets. Composite fonts are classified as Type 0, and are supported by PostScript Japanese printers and by all PostScript Level 2 printers.

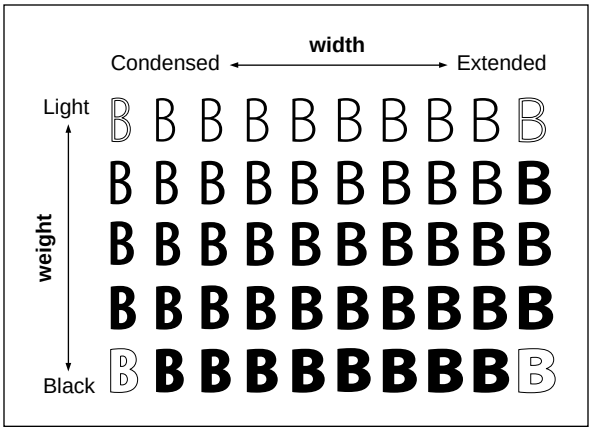
2.3 Multiple Master Type 1 Font Programs

Adobe's multiple master font technology is an extension of the Type 1 font format. These fonts contain two or more complete sets of character outlines, bound in a single font program, from which any number of intermediate instances of the font can be generated. A derived instance of a font requires only a small data structure because the character outlines are shared with the main multiple master font program. This minimizes the storage space needed to generate variations once the basic font program is installed.

Adobe Systems will provide multiple master versions of selected typefaces, including support for multiple master technology in printers, support in application products, and a utility to generate individual fonts derived from the master designs. The derived instances of a typeface are installable in existing applications, and each multiple master font package includes code that makes it backward compatible with installed interpreters.

Typically, a font program provides master designs for light and bold, as well as condensed and expanded typefaces. Users can then generate instances of styles, such as 70% of the boldest style and 55% of the most expanded design. Figure 3 shows the four outline master designs for the letter B (in outline at the four corners) from a multiple master font program. The intermediate characters are instances generated by interpolating between the master designs.

Figure 3 *Outline masters for the letter B from a multiple master font program*



This technology not only gives users greater flexibility and power, but also enables applications to offer the following enhanced features.

- Layout programs create custom-width fonts to help solve text-column justification problems.

- Spreadsheet applications generate custom instances of fonts to do a better job of fitting text into constrained areas.

- Graphic designers create logotypes and advertising designs based on transformations and flexibility.

Another benefit of multiple master font programs is the ability to generate a version of a character that is optically correct for the size at which it will be viewed. In the age of metal type, character design and spacing were adjusted for the point size of the font. Smaller size fonts typically had proportionally wider spacing, heavier stems and serifs, and less contrast between thick and thin strokes than larger size fonts.

This capability was pretty much lost with photo and digital type technologies, but with the new multiple master technology, typefaces can be rendered optically correct for all sizes. To accomplish this, the multiple master font program must contain one set of outlines designed for use at small sizes and one for use at large sizes.

A multiple master font program can also emulate another typeface if that font's metrics are available. Previously, attempts at substituting or scaling one font to match another usually resulted in unsatisfactory results. A specially designed multiple master font can be scaled to match the metrics of another font in a way that preserves consistent stem widths and spacing characteristics, thus preserving legibility in a manner not previously possible. Font substitution can allow users to open a document for which the referenced fonts are not available, and still get high quality typography when viewing or printing that document.

3 Adobe Type 1 Font Packages

Type 1 downloadable font programs are available from a wide variety of sources, including Adobe Systems. These fonts can be installed on the host computer for screen display and for downloading to a printer. They can be downloaded for either a single-print job, semi-permanently (to be RAM resident until the next power cycle), or permanently to a hard disk.

A typical package of downloadable font programs contains:

- One outline font file for each style in the font family

- Screen font files

- One Adobe Font Metrics (AFM) file for each font

- Font utilities: including a PostScript language font and file downloader, an installation utility, and font conversion software. Adobe font packages for the PC include Font Foundry™, which builds .PFM (font metrics file for Windows) files, and bitmapped screen fonts for use with Hewlett-Packard LaserJet® printers.

The outline font file is in a compressed binary format specific to the intended platform. Font programs can be transferred to other platforms, but Adobe recommends that users obtain the same font in the format for the other platform to ease installation and conversion problems and to ensure that the user has both proper documentation and character set and keyboard mapping information. Copies of font programs in alternate formats are offered to registered owners of those fonts for approximately the cost of producing the package.

The screen font files are also in the format required by the platform, although they are no longer shipped with PC font packages. Either the user uses ATM software to generate the screen display for Windows or OS/2, or the Font Foundry utility generates bitmap screen fonts for MS-DOS and specific applications and printers that require bitmap fonts. Macintosh packages include a reduced set of screen fonts; at least one size of these screen fonts must be installed, because the font metrics are available in the screen font suitcase file.

Font metrics files specify character widths that are used for calculating line widths and text placement. They also specify kerning and ligature substitution data, as well as character bounding boxes that can be used for placing accents and determination of clipping regions.

The primary use for metrics is to format text for the screen or printer. For monospaced fonts that emulate typewriter fonts, each character has the same width. With proportionally spaced fonts, each character can have a different width in proportion to its particular design. This results in better quality typography that is easier to read, and the resulting text requires as little as 70% of the space required by a monospaced typeface.

Font metrics are specified for Type 1 fonts in an Adobe Font Metrics (AFM) file. Every style of a font family has a corresponding AFM file. Additional forms of the font metrics might be required and included in the font package, depending on the platform on which the font programs are to be used. For example, a Printer Font Metrics (PFM) file is required for the Windows environment and for the Macintosh environment (where AFM files are rarely used). Metrics for the entire font family are contained in the FOND resource data structure.

4 Font Installation

Font installation generally consists of the user running an installation utility that puts font and metrics files in the appropriate directory. The system might not keep track of installed fonts to provide applications with system calls that can return lists of available fonts for getting font names and presenting font menus.

The essential items an application needs to obtain, either from the system or by itself, are the font name, the location of the screen and printer font file, and the location of the font metrics file. For example, in a MS-DOS® system, font files are deposited in the C:\PSFONTS directory and AFM files in the C:\PSMETRICS directory. An application must parse the AFM files in this directory to get the PostScript language font name.

With the Windows and Macintosh environments, the system provides calls for obtaining lists of available fonts, which makes the process easier. Both systems also have standard locations in which the font and metrics files should reside.

5 User Interface: Font and Style Selection

A user selects fonts from within an application by accessing a font menu. Generally, system calls return the selection of available fonts (the fonts available to the user by virtue of having been installed in the system) to the application. The font itself can reside on the host computer or a font server, or be resident in the ROM, hard disk, or cartridge of the output device.

5.1 Font Names

Font programs are referenced by the name of the font as defined by the **FontName** in the PostScript language font dictionary. When the font is installed in a PostScript language output device, this name is registered in the global font dictionary **FontDirectory**.

The **FontName** remains constant regardless of the platform for which it is intended. Any given operating system might keep track of a font by another name. For example, on the Macintosh, the system registers the name of the Font Resource, which is often not the same as the **FontName**.

Font file names are dependent on the limitations of the platform. For example, with a MS-DOS system on the PC, file names are limited to 8 characters (plus a 3 character extension). Because of the need to represent the style (for example, bold italic), the point size (for bitmap screen fonts), and the device code, the current Adobe convention is to use only the first 2

characters of the 8 to designate the font family name. On the Macintosh platform, up to 29 characters are allowed (limited, for compatibility reasons, by a bug in an early Apple LaserWriter®).

5.2 Font Style Selection

Although some typefaces designed to be used mainly for headlines and display consist of a single style, most typefaces consist of a family of styles related by common design characteristics. For example, a typeface package with a family name such as Garamond, can have a family that consists of Garamond Roman, Garamond Italic, Garamond Bold, and Garamond Bold Italic. Using different typeface, a family is a good way of creating a specific style for a document and adapting that style for different design purposes. Use different styles within a family to articulate and organize various parts of a document, for example, bold for headings and italic for emphasis or differentiation, or to create contrast and variety. Applications need to support this wide variety of typographic style variations as well as provide a good user interface for viewing and selecting the various options.

A large typographic family can consist of seven or eight weights. It can come in widths ranging from ultra-condensed to ultra-expanded. It can also have a roman (upright) and a true italic face (that is, of a cursive design, not an oblique version algorithmically derived from the roman).

Font name and style menu windows should not restrict either the number of variations of a family that can be viewed, nor should the menus limit the length of font names that can be viewed. This is important because in some environments the font name itself is the primary means for users to see which styles are available, and font names are often quite long. For example, a PostScript language **FontName** might be

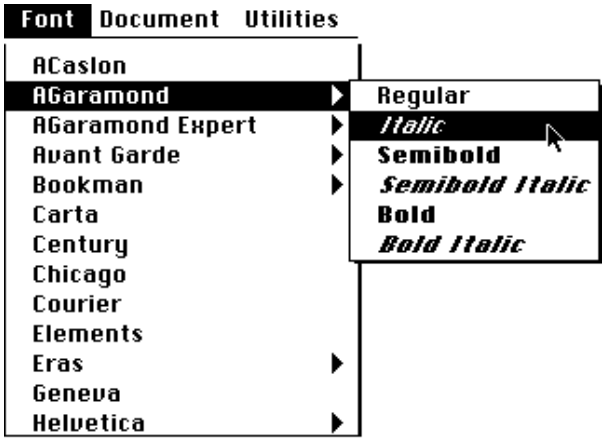
ITCNewCenturySchoolbook-CondensedBoldItalic

where ITC® is the registered trademark name of the vendor who licenses the design, New Century Schoolbook is the typeface family name, and the names following the hyphen describe the style variations.

A preferred way to present font names and style variations is by using a hierarchical menu. Users initially see a menu of alphabetized family names. When a family is selected, the next level menu containing styles appears.

An example of this approach is Adobe Type Reunion™ for the Macintosh, shown in Figure 4. The Macintosh system tracks fonts by a string in the screen font data structure, which is the Resource name. Adobe Type Reunion looks into this data structure to get the PostScript language font name in the Family Style Mapping table to more easily classify the font by family name and style.

Figure 4 *Font selection menu generated by Adobe Type Reunion*



In Figure 4, the first level menu lists family names, and the second level shows the installed style variations of that family. By using a second level menu in this way, font variations can be shown to the user instead of assuming the common variations of roman, italic, bold, and bold italic.

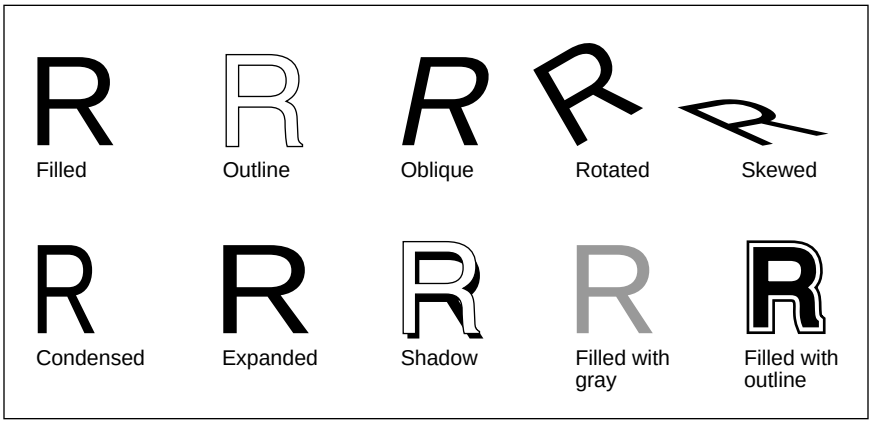
Font and style references should be stored as separate but linked properties. Applications can then allow users to select text and change the font family without changing the style selection. For example, if a paragraph contains one or more italicized words, the user should be able to select a paragraph and change the typeface family used for the whole paragraph, while leaving the style characteristics, such as italic and bold, unchanged.

5.3 Derived Styles and Alternate Characters

An outline font can be mathematically transformed to create a wide variety of alternate styles, as is done in the Macintosh system. Examples of style variations that can be generated from a single roman font include: bold, oblique, horizontally scaled (for fitting text or generating algorithmically condensed and expanded forms), underlined and strike-through text, small capitals, superior and inferior characters, and outline characters.

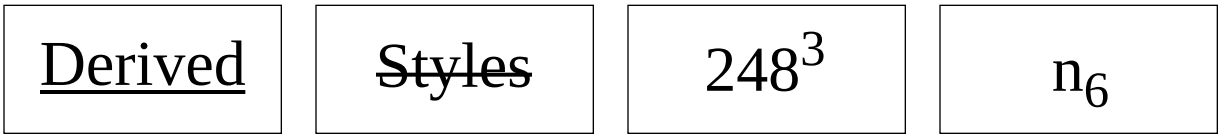
Algorithmic generation of these styles can be useful for extending the choice of styles when there are constraints on memory or disk space. Ideally, however, the true designed form of a character should be substituted wherever possible. This is valuable to the user because a smaller form of a character, such as a superscript figure, does not look optically correct if it is the full figure scaled to a smaller size. Similarly, small caps are designed to be slightly heavier and wider than a full size capital letter scaled to a smaller size. True designed condensed and expanded typefaces are better proportioned than the horizontally scaled characters in the condensed and expanded examples shown in Figure 5.

Figure 5 *Styles derived algorithmically from a single outline character*



For example, an application might have a dialog window allowing the user to specify that a selected number should be a superscript character. The application can determine whether there is an Expert set equivalent of the character and substitute it, rather than requiring the user to change fonts. This feature can be offered for menu selections of small capitals, superscript, and subscript characters available in the Expert fonts. The application can similarly switch fonts for bold and italic variations, for which most font families have true designed fonts.

Figure 6 *Additional derived styles*



Underline *Strike-through* *Superscript* *Subscript*

Figure 6 shows examples of text styles an application can support. Underline and strike-through styles can be generated by drawing lines with PostScript language drawing operators. AFM files provide information on the position and line weight to use for underlining—the position depends on the design of the particular typeface, and the line weight on the character stroke weights.

Small caps, superscripts, and subscripts are examples of characters that appear to be scaled-down versions of the full-sized character. However, to be typographically correct, these characters should be designed to match the color (the relative boldness) of the other characters in the font, and not just algorithmically scaled from the full-size characters. Adobe font programs with the standard Roman character set (defined in Table E.7 of the *PostScript Language Reference Manual, Second Edition*) contain three superior figures 1, 2, and 3, but best results are obtained by fully supporting the Adobe Expert character sets. (See section 9.2.)

6 Screen Display

Text can be displayed in two ways: Either as a character-based display, where the screen display does not exactly represent the printed form or in a WYSIWYG mode. The appropriate approach depends on the application; either can yield good results if handled correctly.

6.1 Character-Based Mode

In character-based mode, a generic font is used to display mainly the content of the text, without attempting to show styles and line and page breaks as they would appear as printed. Although this mode might not be as easy to use as a WYSIWYG system, it need not limit the final quality of a document. High-end typesetting applications have traditionally used this approach to give the user more power and flexibility without requiring as much overhead for the user-interface. However, care must be taken to make the user interface as friendly and as easy to use as possible.

A single size of a bitmapped screen font is generally used for screen display. The display font can be monospaced or proportionally spaced as long as the typeface family name, style variations, and point size are indicated and user-selectable in a properties specification table or menu. One good way to present these properties to the user is to embed mark-up language text into the text of the document. Any manner of differentiating the two texts simplifies the user's job, for example, by using a different color of the font, as was done by the old version of WordPerfect™ for MS-DOS.

6.2 WYSIWYG Mode

With a WYSIWYG approach, ease of implementation can depend on the level of system support for the display of matching screen and printer fonts. This generally requires more development effort and results in a larger size application. It can also mean that the application is less portable because of the heavy use of graphics in the user interface, but these drawbacks can be justified by the enhanced productivity offered to the user.

To achieve WYSIWYG display and printing, an application must use screen fonts that match the metrics of the printer font. This means that character cell widths are equal to the fractional width of the outline character, rounded to the nearest integer number of pixels. Additionally, there must be a file containing the fractional widths, which are used for positioning text on the screen and for determining line and page breaks of the printer font outline characters. These widths can be either in the screen font file data structure, such as for a Macintosh, in the .PFM file in a Windows 3.0 environment, or can be obtained from the AFM file supplied with each font.

Screen display is accomplished by making a system call to display text on the screen (in systems where this is available). For platforms where the ATM software is available and installed, the call will be intercepted by the ATM software, which will display the characters as rasterized from the outline font. By formatting the final PostScript language document using the fractional widths from the metrics file, the printed document will have the same layout as the screen display.

7 Preparing a Document for Printing

Managing font resources for printing a document involves

- determining which referenced fonts are available in the printer
- using the document structuring conventions (DSC) to specify font usage

An application can query a device to determine which fonts are in ROM or cartridge, on the hard disk, or previously downloaded. (Querying is not possible with parallel connections or networks with dumb print spoolers.)

The PostScript printer description (PPD) file defines the font programs installed in a device. The installation utility can modify the PPD file, or the system software or application can keep track of the fonts to reflect any subsequent additions of a font cartridge or fonts to the hard disk.

To specify font usage, PostScript language documents should use comment lines that conform to the DSC. (See the specification in Appendix G of the *PostScript Language Reference Manual, Second Edition*.)

It is essential for an application to determine which fonts are available in the printer so they do not need to be downloaded. A downloaded font can occupy 25 to 50 kilobytes of memory and require 10 seconds or more to transfer the file for an average font. Hence, it is important to use these conventions for all documents created by an application to avoid unnecessary downloading and to help print spoolers and print managers optimize printing.

All font programs referenced by an application should be listed in the document using two DSC comments. Fonts referenced in the document and included in the document file should be listed using the following comment line.

```
%%DocumentSuppliedResources:  < font name list >
```

Font names referenced but not included in the document file should be listed using the following comment line.

```
%%DocumentNeededResources:  < font name list >
```

This list of font names would include both those that the application knows to be ROM or cartridge resident, as well as other font files that the Print Manager must supply (possibly from a font server), or the user must download.

Font changing should be optimized to aid performance because a document can have a large number of font changes on any given page. Font dictionaries and even scaled font dictionaries should be cached to save significant time.

8 Font Downloading and Printing

Once the application determines which fonts it will download or embed in-line, the corresponding disk file must be located.

As explained in section 4, “Font Installation,” an MS-DOS application must parse the AFM files in the C:\PSMETRICS directory for the PostScript language font name to reference in the document; the downloadable font file can then be found by using a .PFB extension and the same name as the AFM file name but in the C:\PSFONTS directory.

On the Macintosh, file names are derived from the PostScript language font name, using uppercase letters and hyphens to aid parsing. The first five letters of the first name are used, followed by the first three letters of any subsequent name components.

A font program can be downloaded prior to sending the document to the interpreter (this is called a pre-loaded font). Alternatively, it can be inserted into the document itself (this is called an in-line font).

Files stored on a disk on either a Macintosh or PC system are in a platform-dependent compressed binary format that must be converted into hexadecimal representation and transmitted in a 7-bit ASCII hexadecimal format to the PostScript interpreter.

The issues involved in managing printer memory and the downloading of fonts are explained in Adobe Technical Note #5048, “Overview of the Generic Text Interface.”

9 Character Sets

Most operating systems have their own standard character sets which is supported for all applications. These character sets are different for each system, and for each application in the case of DOS where there is no standard character set. To support this variety of character sets and encodings, the PostScript language uses a flexible font encoding scheme for linking characters to keyboard codes.

In the Macintosh and Windows environments, the standard character sets are handled by system level PostScript printer drivers. The result is that applications are not free to re-encode fonts to make use of characters which may be in a font but which are not supported by the system.

In a DOS environment, there is no system level support for fonts in general, and hence there are no standard character sets. There also is no system PostScript driver, and each application supplies its own driver. The result is that developers have to do more work to support PostScript output, but they can also provide more flexibility in what characters can be supported. This allows great flexibility for specifying either ASCII or EBCDIC encodings or any other encodings required by the user or system. References to re-encoding in the following sections apply mainly to environments where the application has the freedom to do their own re-encoding.

9.1 Font Re-Encoding

The basic process of re-encoding a font involves making a copy of the font dictionary, replacing the encoding vector in the copied font with a new one, and executing a PostScript language operator to register the new font in **FontDirectory**. The resulting font program is given a new name, which can then be referenced by a PostScript language document.

When re-encoding, either characters can be added to unused places in the array, or the entire encoding array can be redefined. Examples of re-encoding and discussion of techniques can be found in Technical Note #5125, “Roman Font Re-Encoding Issues.”

In the case of the PC and Macintosh, a font with **StandardEncoding** gets re-encoded by the application or system driver to either the MS-DOS, Windows, or Macintosh character set. For the Macintosh, for example, there is a flag set in the screen font data structure that tells the driver that the font uses the **StandardEncoding** and should, therefore, be re-encoded to the Macintosh character set. When the font is used by either ATM or the system’s PostScript driver, the Macintosh character set is built from a combination of the current font and the Symbol font. In the case of the Symbol font and non-standard character sets, a flag is set that tells the driver not to re-encode the font because there is a *font-specific* encoding vector.

9.2 Adobe Character Sets and Encodings

A character set (or glyph complement) refers to the set of all characters included in a PostScript language font program. Not all characters in the font program are encoded or accessible from all applications.

An encoding vector is a PostScript language array that maps character codes (used as indices to the array) to PostScript language character names. These font programs use an encoding whose PostScript language name is **StandardEncoding** (shown in Table E.6 of the *PostScript Language Reference Manual, Second Edition*), which encodes 149 of the characters.

Most text typefaces in the Adobe Type Library use the standard Roman character set, which contains 228 characters. There is also an **ISOLatin1Encoding** vector defined in all PostScript interpreters that encodes 205 characters, including 53 accented characters.

Typefaces that are considered display faces and used mainly for headlines and in advertising have a character set called the Display Character Set (such as the one used for Charlemagne™, Cottonwood™, and other typefaces). This character set is a subset of the standard Roman character set because many characters, such as superior figures, fractions, and math and paragraph symbols, are not used frequently for headlines.

A Type 1 font program can contain any number of characters, but only 255 characters can be encoded at one time. For example, an Adobe font package such as Prestige Elite contains 358 characters, but only 255 of them can be encoded at one time.

9.3 PC Character Sets

Three Adobe font packages, Prestige Elite, Letter Gothic, and Orator, contain the IBM® extended character set, which includes line-drawing characters and a variety of additional symbols. These fonts contain 358 characters, of which only the characters in the **StandardEncoding** set are encoded. Therefore, users can only access these characters if the application supports re-encoding the font. Other typefaces packaged for use in the PC environment utilize the same character set as their Macintosh equivalent, such as the standard Roman character set for most text faces.

9.4 Expert Set and Symbol Fonts

Adobe also offers Expert Set font packages as part of the Adobe Originals series, which features original typeface designs as well as revivals of classic typefaces. This series uses extended character sets for text typefaces.

Using Adobe Garamond™ as an example, the standard font program contains the same character set and encoding as other font packages. An additional font program called Adobe Garamond Expert contains characters such as small caps, old-style figures, additional ligatures, and a variety of other characters regularly used for many publishing and design applications. The Expert font has its own font-specific encoding, which is defined in Appendix E of the *PostScript Language Reference Manual, Second Edition.*

A number of other font programs contain font-specific encoding vectors; examples include Symbol, Carta™, and Sonata™ as well as other math and symbol fonts. The fact that the font program has a font-specific encoding vector — is specified, for example on the Macintosh, in the screen font data structure. This signals the application and driver that the font is not to be re-encoded in the standard way.

9.5 Accented Characters

A common user need is to be able to access the accented characters that are included in most fonts but are not encoded. These characters are particularly important in Europe and other parts of the world, but are frequently requested by domestic U.S. users using foreign words and phrases or publishing translated text.

There are several approaches to supporting accented characters:

- If the system's character set includes accented characters (such as the 36 available in the Macintosh environment), the application does not need to do anything extra to support those specific characters.

- To provide characters not in the system's character set, the application should determine whether the desired character(s) is among the 58 unencoded accented characters in the standard Roman character set, and whether the font has this character set. A DOS-based application can then re-encode the font to include the accented characters. The advantage of using these characters is that the placement of the accent has been done by type designers as opposed to being algorithmically composed.

Note *Early versions of the PostScript interpreter required both components and the composite character to be encoded in the font; more recent versions require only the composite character to be encoded.*

- The application can compose additional accented characters not included in the font. In this case, the base alphabetic character is imaged in the line of text and the position where the next character would appear is saved using the PostScript language command **gsave**. Then the floating accent is positioned over the base character, using the character bounding boxes from the font metrics file, and then the previous position (for the next character) is restored using **grestore** and the text imaging continues.

  Such algorithmic positioning of floating accents may not result in typographically correct text, but is better than not supporting accented characters at all. A nice refinement is for the application to allow the user to adjust the position of the accent and record this offset for future uses in a data table that resides in the application.

Appendix: Changes Since Earlier Versions

Changes since August 11, 1991

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

Changes since May 4, 1991 version

- Section 9.1: Changed the reference from the *PostScript Language Tutorial and Cookbook* to the Technical Note #5125, “Roman Font Re-Encoding Issues.”

Index

R

S

T

U

W