Adobe Type 1 Font Format: Multiple Master Extensions

Adobe Developer Support

14 February 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 LPS5086

Contents

Adobe Type 1 Font Format: Multiple Master Extensions

1 Introduction

This technical note presents the format for multiple master Type 1 font programs. The contents of this document, along with some other minor updates, will be integrated with the *Adobe™ Type 1 Font Format* book and published as version 1.2.

The format extensions discussed in this document are intended to be platform-independent. Information on how to make Macintosh® versions of multiple master fonts is contained in Adobe Technical Note #5087, “Multiple Master Font Programs for the Macintosh”.

2 Multiple Master Overview

The multiple master font format is an extension of the Type 1 font format, which allows the generation of a wide variety of typeface styles from a single font program. This capability allows users and applications unprecedented control over the typographic parameters of fonts used in their documents.

A multiple master font program contains two or more outline typefaces called *master designs*, which describe one or more *design axes*. The master designs that constitute a design axis represent a dynamic range of one typographic parameter, such as the weight or width. This range of styles is defined in a multiple master font program 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 master designs* that are required. The maximum number of master designs allowed is sixteen.

Note *Intermediate designs are not supported in the first version of Adobe Type Manager™ (3.0) to support multiple master font programs.*

A *font instance* consists of a font dictionary derived from the multiple master font program (or from another font instance). It contains a **WeightVector** array with *k* values that sum to 1.0 and which determine the relative contributions of each master design to the resulting interpolated design.

All derived font instances share the **CharStrings** dictionary and **Subrs** array of the main multiple master font program, making it relatively economical to generate a variety of font instances. Multiple master fonts can be made compatible with the installed base of PostScript™ language interpreters by including several PostScript language procedures and a new set of **OtherSubrs** routines in the font program. The procedures include the new **makeblendedfont** operator (see Appendix A), the interpolation procedure **$Blend**, and a new definition of the **findfont** operator (listed in Appendix B).

2.1 Multiple Master Design Space

It is possible to think of the master designs as being arranged in a 1, 2, 3, or 4 dimensional space with various font instances corresponding to different locations in that space. The entries in the **FontInfo** dictionary specify what this space is and where the master designs are located in it. This information is necessary for interactive programs that allow users to create new font instances, and should be included in the font's Adobe Multiple Font Metrics (AMFM) file.

Figure 1 illustrates an example of the design space of a three axis multiple master font. In this example, the axes are *weight*, *width*, and *optical size*. It is recommended that a font program be organized to have the lightest weight, narrowest width, and smallest design size mapped to the origin of the blend space.

Multiple master coordinates are of two types: those which represent the design space and those which represent the blend space.

Design coordinates are integers whose range for a particular typeface is chosen by the designer. They are used in font names and in the user interface for software which creates and manipulates multiple master font programs. The theoretical range for a weight or width axis is from 1 to 999 design units; however a typical typeface, with styles ranging from light to black, might have a dynamic range of from 200 (for light) to 800 units (for black).

Another type of optional axis would be for optical size, in which the character design changes with the point size. The design coordinates for the optical size axis might have a dynamic range of from 6- to 72-point, which represents the practical extremes of sizes for typefaces designed for publishing purposes.

Blend coordinates are normalized values, in the range of 0 to 1, which correspond to the minimum and maximum design space coordinates. They are used by the Type 1 rasterizer because they are more convenient for mathematical manipulations. The linear space of blend coordinates is related to the (potentially) non-linear space of the design coordinates by the **BlendDesignMap** entry (discussed below) in the font dictionary.

Figure 1 *Multiple master typeface design space arrangement*

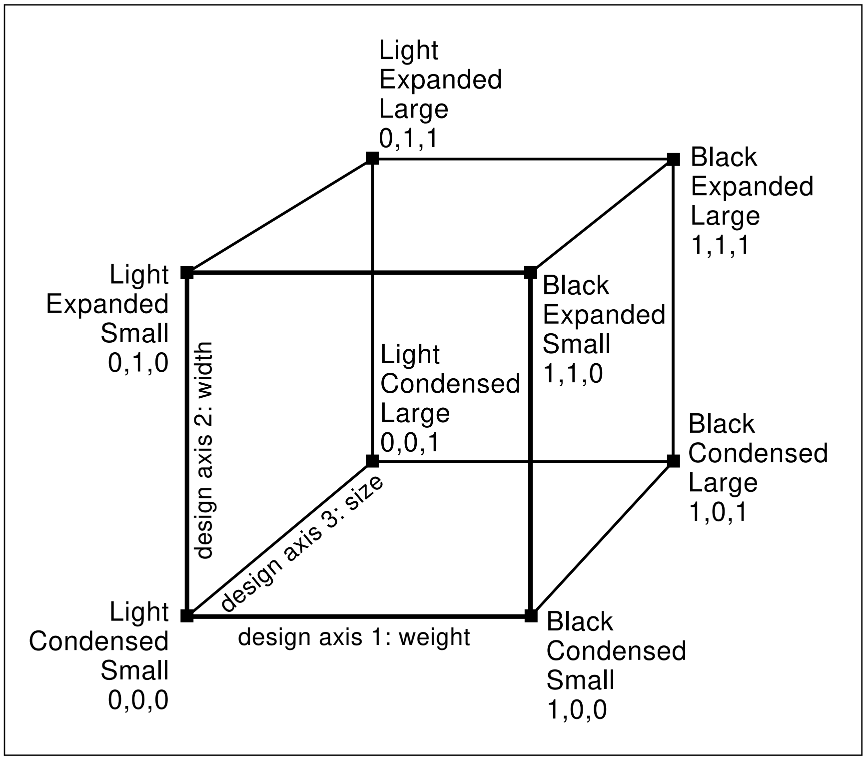
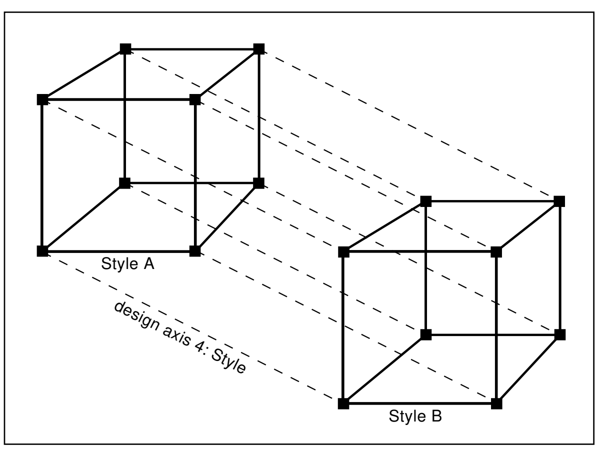


Figure 2 illustrates how a four axis design might be represented. An example of a fourth axis would be having an axis for a typographic style (serif/sans serif) or contrast (high/low: the ratio of thick to thin stem widths). This diagram illustrates that if four axes are defined, sixteen master designs are required. Also, since sixteen is the maximum number of designs allowed, there can be no intermediate designs with four axes.

Figure 2 *Arrangement of the multiple master design space for a four axis font*

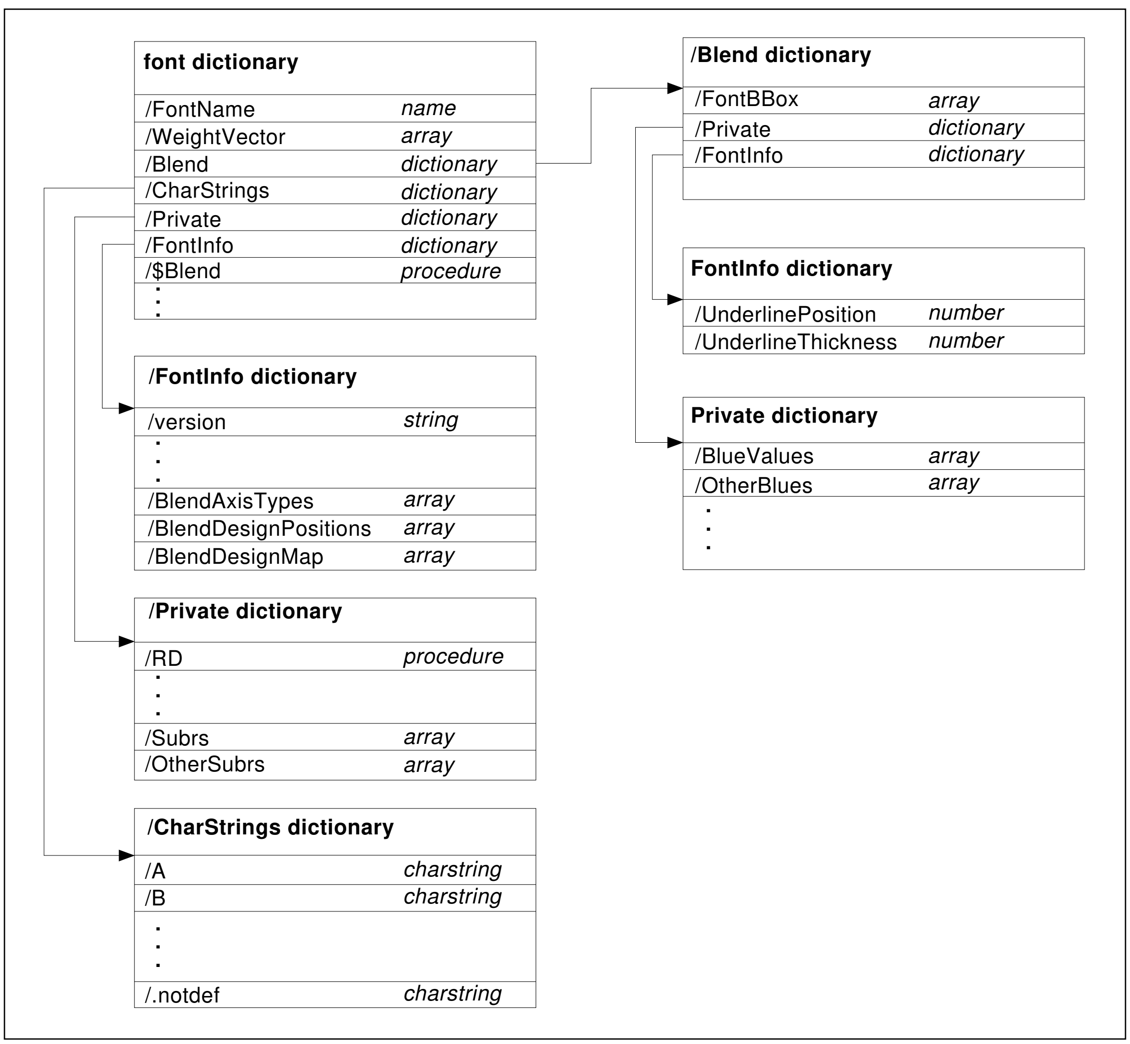


3 Multiple Master Font Programs

Multiple master typefaces may contain from two to sixteen master designs, organized as having from one to four design axes. The maximum number of master designs allowed in a multiple master font 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 allowed number of master designs.

The values used for calculating the weighted average are stored in the font dictionary in an array named **WeightVector**. The multiple master font program, as shipped by the font vendor, can have a default setting for the **WeightVector**; it is recommended that it is set so the default font instance will be the normal roman design for that typeface. Figure 3 shows a diagram of the dictionary organization of a multiple master font program.

Figure 3 *Multiple master font dictionaries*



3.1 The Blend Dictionary

A multiple master font contains a **Blend** dictionary defined in the unencrypted portion of the top level font dictionary. The **Blend** dictionary contains an entry for the blended values for **FontBBox**, plus definitions for two subdictionaries: a **Private** and a **FontInfo** dictionary.

The **Private** blend dictionary (defined in the **Blend** dictionary) will only contain those keywords found in the top level **Private** dictionary which have blended values. These keywords are **BlueValues**, **OtherBlues**, **Std{H,V}W**, **StemSnap{H,V}**, **BlueScale**, **BlueShift**, **Family{Other}Blues** and **ForceBold**. The values for the **Private** subdictionary are expressed as an array with one set of values for each master design; the top level **Private** dictionary contains only single value entries (or set of values, as appropriate to the keyword) which have been blended using the **WeightVector** specified in the font dictionary.

The **FontInfo** subdictionary will contain only keywords found in the top level **FontInfo** dictionary which have *blended* values. The values for the **UnderlinePosition** and **UnderlineThickness** keywords are elements of an array with one value for each master design. It is not necessary to include these entries if they are the same for each design.

The representation of any dictionary entry in the **Blend** dictionary (except **ForceBold**), or in one of the sub-dictionaries under it, is defined by the following recursive rules:

> Let "REP(k)" stand for the Blend dictionary representation for the entry k. If k is a number, then REP(k) is the array of numbers [N_1 ... N_k] that are the values for the entry in the k master designs. If k is an array of n items V_1 ... V_n of any type, then REP(k) is an array of the *n* representations REP(V1) ... REP(Vn). If k is of any type other than *number* or *array*, then REP(k) is k itself.

3.2 Explanation of a Typical Multiple Master Font Program

Example 1 shows a sample multiple master font program.

Example 1:

```
%!PS-AdobeFont-1.0: Myriad 000.009
%%CreationDate: Wed Jul 31 11:43:43 1991
%%VMusage: 69881 80580
15 dict begin
/FontInfo 13 dict dup begin
/version (000.009) readonly def
/Notice (Copyright (c) 1991, 1992 Adobe Systems Incorporated.  All
Rights Reserved.) readonly def
/FullName (Myriad) readonly def
/FamilyName (Myriad) readonly def
```

```
/Weight (All) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -100 def
/UnderlineThickness 50 def
/BlendDesignPositions [[0 0] [1 0] [0 1] [1 1]] def
/BlendDesignMap [[[1 0.00][999 1.00]][[1 0.00][999 1.00]]] def
/BlendAxisTypes [/Weight /Width ] def
end readonly def
/FontName /Myriad def
/Encoding StandardEncoding def
/PaintType 0 def
/FontType 1 def
/WeightVector [0.18 0.07 0.53 0.22 ] def
/$Blend {0.07 mul exch .53 mul add exch .22 mul add add } bind def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/FontBBox{-55.14 -220.84 1148.04 839.18 }readonly def
/Blend 3 dict dup begin
/FontBBox{{-52 -64 -58 -48 }{-212 -216 -224 -222 }{1000 1000 1100
1432 }{828 850 830 867 }}def
/Private 14 dict def
end def
.
% makeblendedfont procedure omitted (see Appendix A)
.
currentdict end
%currentfile eexec
dup /Private 18 dict dup begin
/-|{string currentfile exch readstring pop}executeonly def
/|-{noaccess def}executeonly def
/|{noaccess put}executeonly def
/BlueValues[-11.00 0.00 667.00 685.00 483.48 494.48 650.00 660.56
710.00 720.56 ] def
/OtherBlues[259.46 264.90 -211.20 -200.64 ] def
/BlueScale 0.051208 def
/MinFeature{16 16} |-
/StdHW [67.01] |-
/StdVW [86.14] |-
/StemSnapH [67.01 ] |-
/StemSnapV [86.14 ] |-
/ForceBoldThreshold .57 def
/ForceBold false def
/password 5839 def
3 index /Blend get /Private get begin
/BlueValues[[ -8 -8 -12 -12][ 0 0 0 0][ 664 664 668 668][ 682 682 686
686][ 480 492 480 492][ 488 500 492 504][ 650 650 650 650][ 658 658
662 660][ 692 692 716 716][ 700 700 728 726]] def
/OtherBlues[[ 258 258 262 255][ 263 263 267 262][ -200 -200 -212 -
222][ -192 -192 -200 -212]] def
/BlueScale[ 0.052125 0.052125 0.052125 0.0479583] def
/ForceBold [ false true false true] def
/StdHW [[37 108 39 146 ]] def
/StdVW [[43 155 49 189 ]] def
/StemSnapH [[ 37 108 39 146]] def
/StemSnapV [[ 43 155 49 189]] def
/OtherSubrs
[ {} {} {}
{
```

```
systemdict /internaldict known not
{pop 3}
{1183615869 systemdict /internaldict get exec
dup /startlock known
{/startlock get exec}
{dup /strtlck known
{/strtlck get exec}
{pop 3}
ifelse}
ifelse}
ifelse
} executeonly
{} {} {} {} {} {} {} {} {} {}
{ 4 1 roll $Blend } bind
{ exch 8 -3 roll $Blend exch 5 2 roll $Blend } bind
{ 3 -1 roll 12 -3 roll $Blend 3 -1 roll 9 -3 roll $Blend 3 -1 roll 6
3 roll $Blend } bind
{ 4 -1 roll 16 -3 roll $Blend 4 -1 roll 13 -3 roll $Blend 4 -1 roll
10 -3 roll $Blend 4 -1 roll 7 -3 roll $Blend } bind ]|-
```

This font program begins with an allocation of a dictionary with 15 entries, one of which is the **FontInfo** dictionary. This dictionary defines three elements which define the mapping from the design to the blend coordinate space (see section 2.1, "Multiple Master Design Space").

The font dictionary also contains a new keyword, **WeightVector**, which specifies the contribution of each master design for the current font instance. Its value is an array of *k* elements where *k* is the number of master designs. The elements must sum to 1.0.

The **WeightVector** entry is followed by a **$Blend** procedure which calculates the weighted average of values from the master designs. It uses the values specified by the **WeightVector** array, and is referenced by **OtherSubrs** entries 14 through 18. This procedure should be of the following form for a multiple master font program with *2* master designs:

```
/$Blend {W2 mul add} bind def
```

where W_2 is the second element in the **WeightVector** array. If there are more than two master designs, the procedure should look like:

```
/$Blend { W2 mul exch
   Wi mul exch add
    . . .
   add } bind def
```

where the W_i line (W_i mul exch add) is repeated for i = 3 to *k* (the number of master designs).

After defining the **FontMatrix**, the **FontBBox** value is defined using either a default value for the chosen default master design, or if it is a font instance, it will have a value interpolated by the **makeblendedfont** procedure.

The **Blend** dictionary is then defined, and a **FontBBox** array containing a set of values for each master design is defined in this dictionary. A **Private** sub-dictionary is then created, but no entries are defined until after the blended entries in the encrypted **Private** dictionary have been declared (see section 9, “Adobe Type Manager Compatibility”. (This example does not have a **FontInfo** sub-dictionary under the **Blend** dictionary because the values are the same for all of the master designs.)

The next section of code is the definition of the **makeblendedfont** procedure. This is included in the font for backward compatibility with interpreters in which this operator is not defined (see section 5, “The makeblendedfont Operator”).

All of the remaining code in the example is in the **eexec** encrypted section of the font. This includes the **Private** dictionary, with its blended values (that is, pre-calculated for a specific instance of the font), and the **Private** blend dictionary which was allocated and defined in the **Blend** dictionary, but whose entries are specified in this encrypted portion of the font program. The entries in the **Private** blend dictionary contain arrays with one value for each master design.

A **ForceBold** array may be included in the **Private** subdictionary of the **Blend** dictionary. When this array is present, a new keyword, **ForceBoldThreshold**, must be included in the top level **Private** dictionary. The value for **ForceBoldThreshold** is a number. If the sum of the **WeightVector** elements, for which **ForceBold** is true in the corresponding multiple master font, is greater than or equal to **ForceBoldThreshold** then **ForceBold** is true for the font instance with that **WeightVector**.

The **Private** dictionary continues with the global hint operators, which are the same as for regular Type 1 font programs except that there is one value for each master design. The **OtherSubrs** array for Myriad includes null procedures for the flex mechanism **OtherSubrs**, and includes the **OtherSubrs** code for hint substitution. Next are ten null procedures before **OtherSubrs** entries 14 through18.

4 Multiple Master Keywords

4.1 BlendAxisTypes

(Required.) **BlendAxisTypes** is an array of *n* PostScript language strings where *n* is the dimensionality of the design space and hence the number of axes. Each string specifies the corresponding axis type. In the above 3-axis example, this value would be:

```
/BlendAxisTypes [/Weight /Width /OpticalSize]
```

Note *The keywords Weight, Width, and OpticalSize are reserved for use as axis labels for multiple master font programs. Font developers interested in registering and reserving new labels for additional design axes should write to:*

UniqueID Coordinator
Adobe Developer Support
Adobe Systems Incorporated
P.O. Box 7900
Mountain View, CA 94039-7900

4.2 BlendDesignPositions

(Required.) **BlendDesignPositions** is an array of *k* arrays giving the locations of the *k* master designs in the design space. Each location subarray 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. Table 1 with nine master designs is based on the example shown in Figure 2, with the addition of the first master design which is in the middle of the design space:

Table 1

Design label	*Blend space coordinates*
design 1: regular	.5 .5 .5
design 2: light condensed small	0 0 0
design 3: light expanded small	0 1 0
design 4: black condensed small	1 0 0
design 5: black expanded small	1 1 0
design 6: light condensed large	0 0 1
design 7: light expanded large	0 1 1
design 8: black condensed large	1 0 1
design 9: black expanded large	1 1 1

The **BlendDesignPositions** array would then be:

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

4.3 BlendDesignMap

BlendDesignMap is a required entry consisting of an array of *n* arrays where *n* is the dimensionality of the design space. Each array contains *m* subarrays that describe 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. The order of the subarrays corresponds to the order of design axes in **BlendAxisTypes**.

Example 2:

```
/BlendDesignMap [ [[1 0] [100 .5] [999 1]]  [[1 0] [999 1]]
   [ [6 0] [11 .5] [72 1]] ]
```

The first number in an individual subarray is in design coordinates with a minimum value of 1 and a maximum value of 999. The second number in the subarray is in normalized coordinates, that is, in the range of 0 to 1.

In the above example, the weight and width both range from 1 to 999 in design space, and the third axis, optical size, ranges from 6 to 72 (referring to the range of point sizes for which the typeface can be adjusted for optimal legibility).

5 The makeblendedfont Operator

blendedfontdict weightvector **makeblendedfont** *blendedfontdict´*

This operator creates a font dictionary with blended entries. The *blendedfontdict* argument is a font dictionary of an existing multiple master font; it can be from either the original multiple master font itself, or from an interpolated font instance since any **Blend** dictionary contains all elements needed to derive additional font instances.

The *weightvector* argument is an array of numbers summing to 1.0 to be used as the weights for creating the new font instance. The value of **WeightVector** in *blendedfontdict´* is set to the values in the array *weightvector*. Blended values are calculated for entries in the **Private** and **FontInfo** dictionaries. The result is a font dictionary that can be used as an argument to **definefont**. The resulting dictionary and its contents are still read-write, so the caller of **makeblendedfont** can make further modifications if necessary (such as assigning a **UniqueID**). This **makeblendedfont** operator or procedure will not copy **FID**'s, **UniqueID**'s, or **XUID**'s.

The **makeblendedfont** operator will become a standard feature of the PostScript language and will be defined in **systemdict** as part of the interpreter. For backward compatibility, the downloadable text file for a multiple master typeface should include conditional code (shown in Appendix A) which will check for an existing definition of **makeblendedfont** in either **systemdict**, **shareddict** or **userdict**, and only if none exists will it store a new definition in **shareddict** or **userdict**. If a definition already exists, the font program will reclaim the storage of its own definition by using **save/restore** and use the existing version.

The **Blend** dictionary data structures provide the information needed by the **makeblendedfont** operator (or procedure), without needing to have the specific list of entries to be blended built into the procedure. This allows a single copy of the procedure to be used even if the set of entries to be blended varies in future fonts.

6 CharString Representation

The encoded and encrypted data in the font **CharStrings** and **Subrs** contains the raw (unblended) data from each of the *k* master designs along with calls to the new **OtherSubrs** (see following section on **OtherSubrs**) to blend them together. Each character must be represented by an identical sequence of commands in each master design. The different master designs can differ only in numerical values for their arguments. For example, if the first command in each path for a given character is

```
dx dy rmoveto
```

then the first command for that character would be:

$dx_1\ dy_1\ dx_2{-}dx_1\ \ldots\ dx_k{-}dx_1\ dy_2{-}dy_1\ \ldots\ dy_k{-}dy_1$ `15 callsubr rmoveto`

where dx_i and dy_i come from the i^{th} master design. This makes it possible to use the same **CharStrings** and **Subrs** for all font instances derived from the multiple master font. In this example, $2*k$ values are put on the stack, and **Subrs** entry number 15 calls the **OtherSubrs** entry number 15 which calculates the weighted average for both dx and dy, using the **WeightVector.** This call returns the blended values of dx and dy on the stack which are then used as arguments to **rmoveto**.

Since the font interpreter stack is limited to 24 entries, font programs with four axes may need to call the interpolation procedures in a way that avoids too many elements accumulating on the stack. The limit is effectively 22 items on the stack since the **OtherSubrs** call requires two arguments to pass the **Subr** number and the number of arguments.

For example, the **rrcurveto** operator requires six arguments. If the font has four master designs, then **OtherSubrs** entry 18 cannot be used and the set of arguments must be split into two blend operations. One way to do this is by calling **OtherSubrs** entry 16 twice. In the following example, if there are *k* master designs, there will be *k* sets of:

```
a b c d e f
```

to be blended for each **rrcurveto** operator. If $k = 4$, there would be 26 elements on the stack (including the two extra arguments mentioned above). If these arguments were divided into two calls to **OtherSubrs** entry 16, each of which returns three results, the code would look like:

$a_1\ b_1\ c_1\ a_2{-}a_1 \ldots a_n{-}a_1\ b_2{-}b_1 \ldots b_n{-}b_1\ c_2{-}c_1 \ldots c_n{-}c_1$ 16 **callsubr**
$d_1\ e_1\ f_1\ d_2{-}d_1 \ldots d_n{-}d_1 e_2{-}e_1 \ldots e_n{-}e_1\ f_2{-}f_1 \ldots f_n{-}f_1$ 16 **callsubr** **rrcurveto**

In this example, Subroutine 16 is used to call **OtherSubrs** number 16 as a means of saving space.

7 New OtherSubrs for Multiple Master Font Programs

There are five new entries in the **OtherSubrs** array which are used by multiple master font programs to compute weighted averages using the **WeightVector**. The new entries are numbered 14 through 18, so the necessary number of procedure brackets (“{ }”) must be inserted in the array to fill unused positions.

OtherSubrs 14 through 18 consist of PostScript language code that manipulates the elements on the stack and then calls the **$Blend** procedure (discussed in section 3.2). These routines differ only in the number of results they return. Each must be configured to manipulate the expected number of elements on the stack, which is dependent on the number of master designs, so that they are in the correct order for calling the **$Blend** procedure. Each new **OtherSubrs** requires a new **Subrs** as shown in the example below.

In the summary of **OtherSubrs** calls listed below, *k* is the length of the **WeightVector** array. The **CharStrings** and **Subrs** will use various **OtherSubrs** to create the required weighted averages for various parameters. The font interpreter stack is limited to 24 entries, so some of these **OtherSubrs** may only be useful with fonts having a small value of *k*. (The maximum value of *k* for any font is 16.)

The **OtherSubrs** for multiple master fonts are numbered 14 through 18; the calling sequences are shown below. In each case, *k* is the number of master designs in the font. Each of the following descriptions uses a notation of the form:

$a_1\ a_2{-}a_1\ a_3{-}a_1\ a_4{-}a_1 \ldots a_k{-}a_1$ 14 callsubr => a which indicates the form of the charstrings procedure, and the number of items left on the stack is indicated by the characters to the right of the arrow. The first value is for the first master design, and all subsequent values are expressed as *deltas* relative to the first value. In this example, **Subrs** number 14 is called in the charstring procedure to calculate the weighted average of the input values.

7.1 OtherSubr 14: Input: k values; Result: 1 value

a_1 a_2–a_1 a_3–a_1 a_4–a_1 ... a_k–a_1 14 callsubr => a

Subrs number 14 contains the following call to **OtherSubrs** number 14:

k 14 callothersubr pop

This calling sequence indicates that there are *k* elements on the stack, and the **pop** operator returns the single result.

OtherSubrs entry 14 uses **WeightVector** values to form weighted average of *k* values from the stack. The results are pushed onto the stack. The value of *k* is found from the length of the **WeightVector** array.

7.2 OtherSubr 15: Input: k × 2 values; Results: 2 values

a_1 b_1 a_2–a_1 a_3–a_1 a_4–a_1 ... a_k–a_1 b_2–b_1 b_3–b_1 b_4–b_1 ... b_k–b_1 15 callsubr => a b

Subrs number 15 contains the following call to **OtherSubrs** number 15:

k× 2 15 callothersubr pop pop

OtherSubrs entry 15 uses **WeightVector** values to form two weighted averages, one for the a’s and the other for the b’s.

7.3 OtherSubr 16: Input: k × 3 values; Results: 3 values

a_1 b_1 c_1 a_2–a_1 ... a_k–a_1 b_2–b_1 .. b_k–b_1 c_2–c_1 ... c_k–c_1 16 **callsubr** => a b c

Subrs number 16 contains the following call to **OtherSubrs** number 16:

k×3 16 callothersubr pop pop pop

OtherSubrs entry 16 uses **WeightVector** values to form three weighted averages.

7.4 OtherSubr 17: Input: k × 4 values; results: 4 values

```
a1 b1 c1 d1 a2–a1 ... ak–a1 b2–b1... bk–b1 c2–c1 ... ck–c1 d2–d1... dk–d1
17 callsubr => a b c d
```

Subrs number 17 contains the following call to **OtherSubrs** number 17:

```
k× 4 17 callothersubr pop pop pop pop
```

OtherSubrs entry 17 uses **WeightVector** values to form four weighted averages.

7.5 OtherSubr 18: Input: k × 6 values; Results: 6 values

```
a1 b1 c1 d1 e1 f1 a2–a1 ... ak–a1 b2–b1 ... bk–b1 c2–c1 ... ck–c1 d2–d1
... dk–d1 e2–e1 ... ek–e1 f2–f1 ... fk–f1 18 callsubr => a b c d e f
```

Subrs number 18 contains the following call to **OtherSubrs** number 18:

```
k× 6 18 callothersubr pop pop pop pop pop pop
```

OtherSubrs entry 18 uses **WeightVector** values to form six weighted averages.

8 Multiple Master findfont Procedure

Multiple master font programs from Adobe Systems will include a procedure which will redefine the **findfont** operator in **systemdict**. This is necessary because of the need to generate font instances on-the-fly to satisfy multiple master font references in a PostScript language document.

The PostScript language code necessary to do this is included in Appendix B. The procedure creates all necessary font instances before it calls the standard **findfont** procedure. These instances are only created if the font name conforms to the naming conventions for a multiple master font for the operating system in which they are used. (See Adobe Technical Note #5087, "Multiple Master Font Programs for the Macintosh.")

Two procedures referenced in **findfont**, NormalizeDesignVector and ConvertDesignVector, must be configured for the number of axes and master designs in the font program in which they are used. The NormalizeDesignVector procedure must calculate the normalized equivalent of the design coordinates in the **FontName**, using the values in the **BlendDesignMap** array. These normalized coordinates must be left on the stack for the ConvertDesignVector procedure. This procedure should take the normalized coordinates, generate **WeightVector** values, and leave them on the stack for the **makeblendfont** operator.

9 Adobe Type Manager Compatibility

• The **Blend** dictionary must come after everything in the font dictionary for which blended values can be calculated.

• The keywords **BlendDesignPositions**, **BlendDesignMap**, and **BlendAxisTypes** must be defined before the **Blend** dictionary.

• The **Private** blend dictionary must appear after all elements of the **Private** dictionary for which blended values can be calculated.

Appendix A: The makeblendedfont Procedure

The following is the definition of the **makeblendedfont** operator.

Note *This code, as well as the code in the following appendices, is copyrighted by Adobe Systems Incorporated, and may not be reproduced except by permission of Adobe Systems Incorporated. Adobe Systems Incorporated grants permission to use this code in Type 1 font programs, as long as the code is used as it appears in this document, the copyright notice remains intact, and the character outline code included in such a font program is neither copied nor derived from character outline code in any Adobe Systems font program.*

```
% Copyright 1990, 1991, 1992 Adobe Systems Incorporated.
% All rights reserved.
/shareddict where
{ pop currentshared { setshared } true setshared shareddict }
{ {} userdict } ifelse dup
/makeblendedfont where {/makeblendedfont get dup type /operatortype
eq {
pop false} { 0 get dup type /integertype ne
{pop false} {8 lt} ifelse} ifelse } {true}ifelse
{/makeblendedfont {
8 pop
2 copy length exch /WeightVector get length eq
{ dup 0 exch {add} forall 1 sub abs .01 gt }
{ true } ifelse
{ /makeblendedfont cvx errordict /rangecheck get exec } if
exch dup maxlength dict begin {
false {/FID /UniqueID /XUID } { 3 index eq or } forall
 { pop pop } { def } ifelse
} forall
{ /Private /FontInfo } {
dup load dup maxlength dict begin {
false { /UniqueID /XUID } { 3 index eq or } forall
{ pop pop }{ def } ifelse } forall currentdict end def
} forall
dup /WeightVector exch def
dup /$Blend exch [
exch false exch
dup length 1 sub -1 1 {
1 index dup length 3 -1 roll sub get
dup 0 eq {
pop 1 index {/exch load 3 1 roll} if
/pop load 3 1 roll
```

```
} {dup 1 eq {pop}
{2 index {/exch load 4 1 roll} if
3 1 roll /mul load 3 1 roll } ifelse
1 index {/add load 3 1 roll} if
exch pop true exch} ifelse
} for
pop { /add load } if
] cvx def
{2 copy length exch length ne {/makeblendedfont cvx errordict /
typecheck get exec}if
0 0 1 3 index length 1 sub {
dup 4 index exch get exch 3 index exch get mul add
} for
exch pop exch pop}
{{dup type dup /arraytype eq {
  pop 1 index /ForceBold eq {
  5 index 0 0 1 3 index length 1 sub {
  dup 4 index exch get {2 index exch get add } {pop} ifelse
  } for exch pop exch pop
  2 index /ForceBoldThreshold get gt 3 copy} {
{length 1 index length ne { pop false } {
true exch { type dup /integertype eq exch /realtype eq exch or and }
forall
} ifelse }
2 copy 8 index exch exec {pop 5 index 5 index exec}
{exch dup length array 1 index xcheck { cvx } if
dup length 1 sub 0 exch 1 exch {
dup 3 index exch get dup type /arraytype eq {
dup 10 index 6 index exec {
9 index exch 9 index exec} if } if 2 index 3 1 roll put
} for exch pop exch pop
} ifelse 3 copy
1 index dup /StemSnapH eq exch /StemSnapV eq or {
dup length 1 sub {dup 0 le { exit } if
dup dup 1 sub 3 index exch get exch 3 index exch get 2 copy eq {
pop 2 index 2 index 0 put 0 } if le {1 sub}
{dup dup 1 sub 3 index exch get exch 3 index exch get
3 index exch 3 index 1 sub exch put
3 copy put pop
2 copy exch length 1 sub lt {1 add} if} ifelse} loop pop
dup 0 get 0 le {
dup 0 exch {0 gt { exit } if 1 add} forall
dup 2 index length exch sub getinterval} if } if } ifelse put }
{/dicttype eq {6 copy 3 1 roll get exch 2 index exec}
{/makeblendedfont cvx errordict /typecheck get exec} ifelse
} ifelse pop pop } forall pop pop pop pop }
currentdict Blend 2 index exec
currentdict end
} bind put
/$fbf {FontDirectory counttomark 3 add -1 roll known {
cleartomark pop findfont}{
] exch findfont exch makeblendedfont
dup /Encoding currentfont /Encoding get put definefont
} ifelse currentfont /ScaleMatrix get makefont setfont
} bind put } { pop pop } ifelse exec
```

Appendix B: Multiple Master findfont Procedure

The following **findfont** procedure can be included in a multiple master font programs to generate font instances for all references to multiple master font instances in a PostScript language document. Two procedures referenced below, NormalizeDesignVector and ConvertDesignVector, need to be configured for the number of axes and master designs in the font program they are used in (see section 8, "Multiple Master findfont Procedure")..

```
% Copyright 1991, 1992 Adobe Systems Incorporated.
% All rights reserved.

  /$mmff_origfindfont where {
  pop save { restore } { pop pop }
} { {} { def } } ifelse
/setshared where { pop true } { false } ifelse
/findfont where pop dup systemdict eq {
pop { currentshared {{}} { true setshared { false set-
  shared } } ifelse shareddict
} {{} userdict } ifelse begin
} { begin { currentdict scheck } { false } ifelse {
currentshared {{}} { true setshared { false setshared }
  } ifelse
} { {} } ifelse } ifelse
/$mmff_origfindfont /findfont load 3 index exec
/findfont {
dup FontDirectory exch known
{ dup FontDirectory exch get /FontType get 3 ne}
{ dup SharedFontDirectory exch known
{ dup SharedFontDirectory exch get /FontType get 3 ne}
{ false} ifelse} ifelse
{$mmff_origfindfont} { dup dup length string cvs (_)
  search {
cvn dup dup FontDirectory exch known exch SharedFontDi-
  rectory exch known or {
$mmff_origfindfont begin pop
[ exch { (_) search  { { cvr } stopped { pop pop } {
exch pop exch } ifelse
} { pop exit } ifelse } loop false /FontInfo where {
pop FontInfo /BlendAxisTypes 2 copy known {
```

```
get length counttomark 2 sub eq exch pop
} { pop pop } ifelse } if {
NormalizeDesignVector
ConvertDesignVector
] currentdict exch makeblendedfont
2 copy exch /FontName exch put
definefont} { cleartomark $mmff_origfindfont } ifelse
  end
} { pop pop pop $mmff_origfindfont } ifelse
} { pop $mmff_origfindfont } ifelse } ifelse
} bind 3 index exec
/SharedFontDirectory dup where { pop pop } { 0 dict 3
  index exec } ifelse
end exec pop exec
```

Index