XEROX

Font Interchange Standard

Xerox
System Integration
Standard

FONT INTERCHANGE STANDARD

XEROX

XNSS 238512, Version 1.0
December 1985

Notice

This *Xerox System Integration Standard* describes the Font Interchange Standard which defines a digital representation for interchange of fonts and font metrics.

This document is being provided for informational purposes only. Xerox makes no warranties or representations of any kind relative to this document or its use, including the implied warranties or merchantability and fitness for a particular purpose. Xerox does not assume any responsibility or liability for any errors or inaccuracies that may be contained in the document, or in connection with the use of this document in any way.

The information contained herein is subject to change without any obligation of notice on the part of Xerox.

This document was produced using the Interpress 3.0 standard. Text and graphics were created on the Xerox 8010 and 6085 Professional Computer Systems using the ViewPoint software. The figure in Chapter 2, Font Descriptions, was created using Interpress 3.0 instructions directly. The camera-ready copy was produced on a very high-resolution Interpress laser printer at Xerox.

The document is available in Interpress, using the Commercial Set, for printing on any of the Xerox Interpress network laser printers including the Xerox NS 8000 LaserCP, 8044, 4050, 8700, and 9700.

Printed in U.S.A. Publication number: 610P50677

PREFACE

This publication is one of a family of publications that collectivcely describe the standards underlying Xerox Printing Systems and the Xerox Network Systems (XNS) Architecture.

A *font* is a vector of characters, each character defined by a mask operator for imaging its graphical shape, and by *metrics* for determining its spacing. Besides the metrics used to determine spacing, there are also other metrics of the font, other measurements or other properties useful to some users of the font. This Font Interchange Standard defines a digital representation for interchange of fonts and font metrics.

The *Interpress Electronic Printing Standard*, XNSS 048601, defines a digital representation for interchange of material to be printed or otherwise imaged. This Font Interchange Standard is defined in terms of that Interpress standard. In particular, this Font Interchange Standard utilizes the facilities of Interpress to enable describing fonts independently of any particular printing or imaging device.

This Font Interchange Standard does not specify which characters are to be present in a font; the assignment of their character codes, i.e., of their indices into the font vector; the partitioning of fonts into particular character sets or collections; the naming of fonts or font files; or the protocols for transporting font files. These topics are or may become the subject of other Xerox System Integration Standards.

Comments and suggestions on this document and its use are encouraged. Please address communications to:

Xerox Corporation
Printing Systems Division
Printing Systems Administration Office
701 South Aviation Boulevard
El Segundo, California 90245

TABLE OF CONTENTS

1. INTRODUCTION

A *font* is a vector of characters, each character defined by a mask operator for imaging its graphical shape, and by *metrics* for determining its spacing. The characters in a font are usually designed to appear consistent with each other when printed. Besides the metrics used to determine spacing, there are also other metrics of the font, other measurements or other properties useful to some users of the font. This Font Interchange Standard defines a digital representation for interchange of fonts and font metrics.

The *Interpress Electronic Printing Standard*, XNSS 048601, defines a digital representation for interchange of material to be printed or otherwise imaged. This Font Interchange Standard is defined largely in terms of that Interpress standard. In particular, this Font Interchange Standard utilizes the facilities of Interpress to enable describing fonts independently of any particular printing or imaging device. This standard also shares the terminology and the notational conventions of the Interpress standard.

As in the Interpress standard, occasional paragraphs of fine print, like this, are used in this Font Interchange Standard for fine points, information auxiliary to the standard itself.

Chapter 2 of this document reiterates and extends the definition of an Interpress data structure called a FontDescription. Chapter 3 uses that definition to in turn define font and metric masters, the main vehicles of font interchange. Chapter 4 deals with various related pragmatic issues. Appendix A lists references, and Appendix B gives examples of font and metric masters.

2. FONT DESCRIPTIONS

The Interpress standard defines a *FontDescription*, an Interpress universal property vector with certain properties. A FontDescription is the input to the Interpress MAKEFONT operator, which produces a *Font*. This chapter briefly reiterates that definition, and defines additional properties of a FontDescription. This definition will be used in Chapter 3 to define font masters and font metric masters.

A FontDescription has the following property names and values. The Interpress standard defines only the properties *characterMasks*, *characterMetrics*, *transformation*, and *substituteIndex*, and only those properties are used directly by an Interpress printer; all other properties are provided for the use of Interpress creators or other users.

characterMasks:
Property vector of composed character mask operators, as defined in the Interpress standard, and briefly described in §2.1 of this standard.

characterMetrics:
Property vector of CharacterMetrics universal property vectors, containing metric information for each character, as defined in the Interpress standard, and further elaborated in §2.2 of this standard.

fontMetrics:
Universal property vector, containing metric information about the font as a whole, defined in §2.3.

transformation:
Transformation, transforming all character masks and font and character metrics in this FontDescription into the character coordinate system, as defined in the Interpress standard.

substituteIndex:
Cardinal, the index of a character within this font to be substituted for any character not present in the *characterMetrics* vector, as defined in the Interpress standard.

pixelArrayTransformationUses:
Vector of transformations, describing the scales and rotations of the pixel arrays used in defining the character masks in this FontDescription. Each element in the vector is a transformation from the pixel array coordinate system to the character coordinate system, as those coordinate systems are defined in the Interpress standard. For example, the transformation ⟨1/50 SCALE⟩ describes a body size of 50 pixels and normal scan orientation within the character coordinate system. A transformation ⟨0 SCALE⟩ indicates that no pixel arrays are used in the masks. Notice that this information, combined with an easy transformation from the pixel array coordinate system to the Interpress coordinate system (i.e., combined with a resolution and scan orientation of a printing device), indicates the sizes and rotations of fonts that can be easily produced with these masks; to continue the preceding example, a transformation of ⟨1/50 SCALE⟩, combined with a printing

resolution of 300 spots per inch and normal scan orientation, indicates that these masks will produce 50/300 inch, portrait orientation characters.

easy:
Vector of transformations, describing the sizes and rotations of the font that can be supplied easily by the supplier of this FontDescription. Although Interpress allows character operators to be called with arbitrary transformations, font or device limitations may prevent precise rendition of characters in all cases. The *easy* vector suggests character sizes and orientations that can be supplied with greatest fidelity. Each element in the vector is a transformation from the character coordinate system to the Interpress Coordinate System, as those coordinate systems are defined in the Interpress standard, that describes a combination of rotation and scaling that can be supplied easily; for example, the transformation ⟨254/72000 SCALE⟩ describes a 10/72-inch body size oriented to point up in the normal viewing orientation. A transformation ⟨0 SCALE⟩ indicates that the font can be supplied easily with an arbitrary transformation.

characterCollections:
Vector of identifiers, each indicating a collection of characters included in its entirety in this FontDescription.

fontVersion:
Identifier, the version of this FontDescription. For any two FontDescriptions with equivalent *name* and *fontVersion*, the two FontDescriptions may be assumed to be logically equivalent.

name:
Vector of identifiers, the *universal name* of the font, as defined in the Interpress standard.

The Xerox Print Service Integration Standard further defines the structure of certain font names, particularly those whose first, universal identifier is *Xerox*.

formatVersion:
Cardinal = 5, the version number of this FontDescription format. This number will be incremented with each change, however slight, to this FontDescription definition.

format:
Identifier = *FontDescription*, identifying the format of this FontDescription vector.

2.1 CharacterMasks

The *characterMasks* element of a FontDescription is a property vector of composed character mask operators, as defined in the Interpress standard. Each property name in this vector is a Cardinal, the *character index* or code. Each corresponding property value is an operator which generates the character's mask, thus defining the shape of the character.

Neither Interpress nor this Font Interchange Standard establish particular conventions for the correspondence between character indices and character symbols or shapes. In this way, a font may represent an arbitrary "character code mapping," i.e., an arbitrary mapping from character indices to symbols or shapes.

The Xerox Character Code Standard, XNSS 058512, defines one particular comprehensive mapping from character indices to symbols or shapes.

For purposes of defining its mask or masks, a character mask operator may potentially call upon any or all of the graphic facilities of Interpress, including pixel arrays and geometrically defined masks. The operator is defined so that the *transformation* element of the FontDescription transforms it into the Interpress character coordinate system, with a nominal body size of one unit and portrait orientation; the actual size, orientation, and placement of the mask is controlled by the current transformation at the time of imaging.

A character present in the *characterMetrics* vector but not in the *characterMasks* simply has no mask, i.e., it is a space.

2.2 CharacterMetrics

The *characterMetrics* element of a FontDescription is a property vector of CharacterMetrics universal property vectors, as defined in the Interpress standard. Each property name in the *characterMetrics* vector is a Cardinal, the *character index* or code. Each corresponding property value is the CharacterMetrics property vector for the character. A CharacterMetrics property vector contains metric information about a single character.

The CharacterMetrics property names and values are given following. The Interpress standard defines only the properties *escapementX*, *escapementY*, *amplified*, and *correction*, and only those properties are used directly by an Interpress printer; all other properties are provided for the use of Interpress creators or other users. All dimensions are defined so that the *transformation* element of the FontDescription transforms them into the Interpress character coordinate system, with a nominal body size of one unit and portrait orientation.

escapementX:
Number, the x component of the *escapement* of the character, as defined in the Interpress standard. If the *escapementX* property is not present, its value may be assumed to be zero.

escapementY:
Number, the y component of the *escapement* of the character, as defined in the Interpress standard. If the *escapementY* property is not present, its value may be assumed to be zero.

The Xerox Character Code Standard specifies the *escapements* for a certain few characters in fonts coded according to that standard.

amplified:
Cardinal, indicating whether the character's escapement can be *amplified* to achieve justification, as defined in the Interpress standard. The value of *amplified* is non-zero, e.g., 1, if the character is amplifying, and 0 if the character is non-amplifying. If the *amplified* property is not present, its value may be assumed to be zero, i.e., non-amplifying.

correction:

Cardinal, indicating what operator, if any, is called to correct spacing, as defined in the Interpress standard. The value 0 indicates CORRECTMASK called, 1 indicates CORRECTSPACE, and 2 indicates no operator called. If the *correction* property is not present, its value may be assumed to be zero, i.e., CORRECTMASK.

leftExtent, rightExtent, descent, ascent:

Number, describing the horizontal and vertical extent of the character mask, as measured from the character *origin* defined in the Interpress standard. The numbers are all signed, and represent distances in the leftward, rightward, downward, and upward directions from the origin respectively; see Figure 2.1. For example, the *descent* of a single quote character (') will usually be negative; similarly, the *leftExtent* of any character whose mask is located entirely to the right of its origin will be negative. All four numbers are zero if the character contains no mask, i.e., if it is a space.

Note that the mask is considered to extend only to those areas where it causes the page image to be modified. In particular, in the case of a mask defined by a pixel array, sample values of 1 identify pixels that are part of the mask; but sample values of 0 identify pixels that are not part of the mask, and are therefore not part of the mask's extent.

centerX, centerY:

Number, the coordinates of the *optical center* of the character, as measured from the character *origin*. The optical center is a point about which the character appears optically balanced.

kerns:

Vector of vectors, suggesting character-pairwise spacing adjustments based upon various *successor* characters to this character and in the same font. Each element of the *kerns* vector is itself a three-element vector with these elements:

successor (index 0):

Cardinal, the index of a successor character.

kernX (index 1), *kernY* (index 2):

Number, the signed amount to add to this character's *escapement* to space this character to the successor character.

Note that the font itself can directly incorporate kerning in the traditional sense, wherein the character shape extends beyond its body. It is only when the spacing between two adjacent characters is to be adjusted based upon the identity of both characters, as in pairwise kerning, that this *kerns* vector is potentially required.

ligatures:

Vector of vectors, suggesting ligature substitutions based upon various *successor* characters in the same font. Each element of the *ligatures* vector is itself a two-element vector with these elements:

successor (index 0):

Cardinal, the index of a successor character.

ligatureCharacter (index 1):

Cardinal, the index of a ligature character to use in place of the two-character sequence consisting of this character and the successor character. Note that the ligature character can also have a ligature, thus allowing arbitrarily long ligature sequences to be specified.

The Xerox Character Code Standard effectively specifies the *ligatures* for fonts coded according to that standard.

superscriptX, subscriptX:
Number, suggesting x coordinates relative to this character's origin at which to place the origins of trailing superscript and/or subscript characters.

When these values are not given, it is usually effective to horizontally space superscripted and subscripted text according to the regular escapementX values of the characters.

superscriptY, subscriptY:
Number, suggesting y coordinates relative to this character's origin at which to place the origins of superscript and/or subscript characters. The corresponding font metrics, if present, provide defaults for these character metrics.

2.3 FontMetrics

The *fontMetrics* element of a FontDescription is a universal property vector, containing metric information about the font as a whole. The *fontMetrics* element is not defined in the Interpress standard, and the *fontMetrics* element is not used by Interpress printers; *fontMetrics* is provided for the use of Interpress creators and other users.

The *fontMetrics* properties are as follows. All dimensions are defined so that the *transformation* element of the FontDescription transforms them into the Interpress character coordinate system, with a nominal body size of one unit and portrait orientation.

typefaceName:
Vector of Cardinals, a string of ISO 646 7-bit Coded Characters, indicating the name of the font typeface, or indicating the name of a well-known face to which the font is similar, in a human-sensible form, e.g., "Times Roman."

minimumScale, maximumScale:
Number, suggesting the minimum and maximum positive scale transformations for which this font looks good. Although Interpress allows character mask operators to be called with arbitrary transformations, typographic considerations suggest that some fonts will look good over only a limited range of sizes. Each number, *scale*, defines a transformation ⟨*scale* SCALE⟩ from the character coordinate system to the Interpress Coordinate System; for example, the number 254/72000 indicates the transformation ⟨254/72000 SCALE⟩, corresponding to a suggested minimum or maximum size of 10/72 inches.

capHeight:
Number, the height of upper-case characters in the font, measured upward from the baseline. Generally pertains only to Latin alphabets.

In Latin alphabets, the *baseline* is a line just under the bottom of non-descending characters. The Interpress standard establishes the convention that for Latin alphabets, the origins of characters are on this baseline.

xHeight:
Number, the height of lower-case characters in the font, measured upward from the baseline. Generally pertains only to Latin alphabets.

lowerCaseDescent:
Number, the maximum descent of lower-case characters in the font, measured downward from the baseline. Generally pertains only to Latin alphabets.

slant:
Number, the slant of the characters defined by the character mask operators, given in degrees to the right of the vertical. For example, the slant of a Times Roman font would be 0 degrees; the slant of a Times Italic might be 12 degrees. This value can be used to position accent marks.

superscriptY, subscriptY:
Number, suggesting y coordinates relative to the baseline at which to place the origins of superscript and/or subscript characters. These values may be overridden for specific characters by specific character metrics.

When these subscript values are not given, it is usually effective to take *superscriptY* = the cap. height of this font minus the x height of the superscript font; and to take *subscriptY* = minus the lower case descent of this font.

underlineOffset:
Number, suggesting the distance downward from the baseline where the center of a single underline bar should appear.

underlineThickness:
Number, suggesting the thickness of the desirable underline bar.

doubleUnderlineUpperOffset, doubleUnderlineLowerOffset:
Number, suggesting the distances downward from the baseline where the centers of double underline bars should appear.

doubleUnderlineThickness:
Number, suggesting the thickness of each of the desirable double underline bars.

strikeOutOffset:
Number, suggesting the distance above the baseline where the center of a strike-out bar should appear.

When a *strikeOutOffset* value is not present, it is usually effective to take *strikeOutOffset* = (the x height of the font + the thickness of the strike-out bar)/2.

strikeOutThickness:
Number, suggesting the thickness of the desirable strike-out bar.

When a *strikeOutThickness* value is not present, it is usually effective to take *strikeOutThickness* to be the same as the thickness of the en-dash in the font, if known.

Font users may wish to retain additional, redundant information about fonts, such as the maximum bounding box of all characters in the font, or the maximum escapement, or whether all characters in the font have the same escapements. This information need not be represented explicitly here in the font-wide metric information, as it can be derived by inspecting all the character metrics for the font.

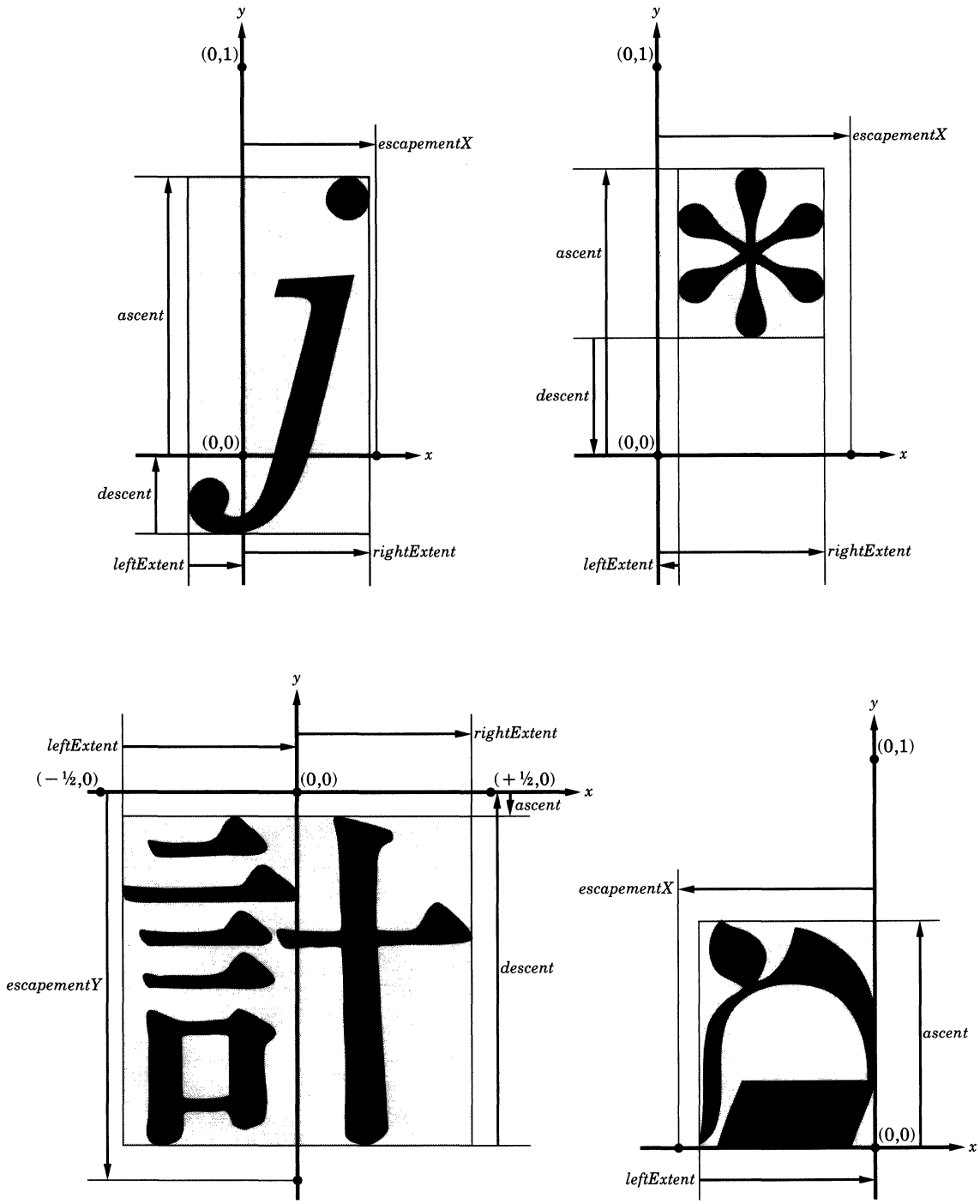

Figure 2.1 **Character metrics**

3. FONT AND METRIC MASTERS

This chapter defines font masters and font metric masters, which are the main vehicles of font interchange. The definition of these masters is given largely in terms of the FontDescription property vector defined in Chapter 2.

3.1 Font masters

A *font master* is an Interpress master, defined and encoded according to the Interpress standard; whose preamble, when executed, leaves on the Interpress machine stack only one or more of the FontDescription vectors described in Chapter 2; where at least the *characterMasks*, *characterMetrics*, *transformation*, *pixelArrayTransformationUses*, *name*, *formatVersion* and *format* elements must be present in each such FontDescription vector. An instructions body may be present in the master. No page bodies or other node content may be present.

3.2 Metric masters

A *font metric master* is an Interpress master, defined and encoded according to the Interpress standard; whose preamble, when executed, leaves on the Interpress machine stack only one or more of the FontDescription vectors described in Chapter 2; where at least the *characterMetrics*, *transformation*, *name*, *formatVersion* and *format* elements must be present in each such FontDescription vector. An instructions body may be present in the master. No page bodies or other node content may be present.

This Font Interchange Standard, in conjunction with the Interpress standard, defines the format of font masters and font metric masters. This chapter describes various additional, useful techniques that are available within that environment.

4.1 Interpress sets

Many devices are able to handle some, but not all, of the capabilities of the Interpress language. When using Interpress as the basis for a Font Interchange Standard, many devices are similarly able to handle some, but not all, of the capabilities of the Font Interchange Standard implied by its use of the Interpress language.

The Interpress standard defines a printing instruction, *set*, which identifies the set of Interpress types and primitives used in a particular master. Thus a font or metrics master can be defined using only some defined Interpress set; and the *set* instruction can be used to communicate along with the master that it uses only such a set. Font and metric masters are to use one of the following defined sets.

The *FIS/simpleFontMetrics* set allows only Numbers, Identifiers, and the primitive operators MAKEVEC, MAKEVECLU, ROTATE, SCALE, SCALE2, and CONCAT. No Interpress encoding notations are allowed. Minimum values for size limits are as specified in the Interpress standard.

The *FIS/bitmapFonts* set consists of the Interpress base language, excluding control, test, and arithmetic operators; plus transformations whose rotation components are integer multiples of 90 degrees, current position operators, FINDDECOMPRESSOR, MAKEPIXELARRAY, MASKPIXEL, and MASKCHAR. Minimum values for size limits are as specified in the Interpress standard.

The *FIS/publicationFonts* set consists of the Interpress *publication* set, but excluding the FINDFONT operator.

The *FIS/professionalGraphicsFonts* set consists of the Interpress *professional graphics* set, but excluding the FINDFONT operator.

Note that an Interpress master's preamble can actually execute only BASE and WEAKIMAGE operators, regardless of the master's declared set, but can make composed operators containing arbitrary IMAGE operators up to the limits of its set.

4.2 Composite characters

Within a font, a single character index or character code may identify a composite character, i.e., a character whose shape is the composite of two or more other component characters within the font. For example, a single accented character might be composed of a base character and an accent character in some specified arrangement; (è) might be composed of the base character (e) and the accent (`). For another example, within some font there might be no ligatured rendering of some ligature character; in this case, the font might choose to render the ligature using its component characters; the ligature (ﬀ) might be rendered as the character pair (ff).

There are (at least) two ways within the Font Interchange Standard to create appropriate character operators for such composite characters. One way is to directly incorporate into the mask operator for the composite character a duplicate of the combined masks for the component characters. This is simple and fast, but bulky.

Another method is to have the composite character's mask operator call upon the mask operators of the component characters, with interleaved positioning operations as necessary, to create the composite shape. The Interpress operators SHOW, *CHAROP, and MASKCHAR are defined so as to permit this. This is slower and more complex, but more compact because it avoids duplicating the masks.

Composite characters are achieved by having the composite character's mask operator use the Interpress operator MASKCHAR to obtain and execute each of the component characters' masks. For example, this composed mask operator might be used for the character (è):

```
{                          --begin composed operator body--
 101                       --= 145₈, character index for (e)--
 MASKCHAR                  --do mask for (e)--
 3 0 TRANSLATE CONCATT     --adjust position slightly--
 96                        --=140₈, character index for (` )--
 MASKCHAR                  --do mask for (` )--
} MAKESIMPLECO             --end body, make composed mask operator for (è)--
```

In order that the character mask may be easily recognized as a composite, its operator should generally be coded as shown in the example, as a sequence of calls to MASKCHAR, optionally interleaved by calls to TRANSLATE and CONCATT.

Notice that it is only the masks of the composite's component characters which are composed in this way. The metrics for the composite are defined alone in the usual manner, with no reference to the metrics of other characters.

4.3 Fragmented fonts

It is often convenient to describe a font as consisting of multiple font fragments, with each fragment describing some characters and represented by a separate master. For example, a font might be fragmented into arbitrary character sets, each set in a separate master, and the complete font defined in a third master to be a composite of the sets. Or it might be desired to have a "pi" (typeface independent) character collection in one master, and to

combine it with numerous different typeface-specific character collections, each in a separate master, to produce numerous different, complete fonts.

A font fragment can be represented by a master whose preamble, when executed, leaves on the stack a FontDescription vector. The *characterMetrics* element generally must be present in the resulting FontDescription; the *characterMasks* must generally be present as well if the fragment is to to be combined to produce a font master and not just a metric master. Another master can insert the font fragment into itself, using *sequenceInsertFile*, and extract the desired character information. Generally the extracted character information from one source is merged with other character information using MERGEPROP; note that MERGEPROP enforces a priority between the two inputs for any character index defined in both inputs.

For example, assume the local files with names "charSet0" and "charSet1" are masters containing the character masks and metrics for two different character sets. Then the following code in a third master will produce character masks and metrics vectors for the merger of the two sets.

```
sequenceInsertFile "charSet0"  --gets preamble body--
DOSAVESIMPLEBODY               --perform preamble, get FontDescription for char. set 0--
DUP                            --duplicate FontDescription--
characterMasks GETP            --get character masks from char. set 0--
EXCH                           --swap with dupe. of FontDescription--
characterMetrics GETP          --get character metrics from char. set 0--

sequenceInsertFile "charSet1"
DOSAVESIMPLEBODY               --get FontDescription for char. set 1--
DUP
characterMetrics GETP          --get metrics from char. set 1--
EXCH
characterMasks GETP            --get masks from char. set 1--

3 2 ROLL                       --get metrics from both sets on top--
MERGEPROP                      --merge metrics from both sets--
3 2 ROLL                       --get masks from both sets on top--
MERGEPROP                      --merge masks from both sets--
```

The preceding code produces character masks and metrics vectors for the merger of the two input character sets. Here MERGEPROP dictates that char. set 1 has priority in case of characters defined in both inputs. This code would usually appear in the preamble of a font master, and be followed by additional code to create a complete FontDescription.

4.4 Raster encoding and compression

In defining a character operator, it is generally necessary to call upon the Interpress Environment for pixel array (or *raster*) decompression operators, and for color operators. Within the Xerox portion of the Interpress Environment, the *Raster Encoding Standard*, XNSS 178506, defines the recognized decompression and color operators, and the general extent of support for raster images.

4.5 Master and font compression

There are a variety of available techniques for effectively reducing the size of a font or metric master, and/or of a Font produced by the FontDescription produced by the master. Several of these compression techniques have already been mentioned in this section. Composite character techniques allow re-use of certain character masks. Composite fonts allow re-use of certain font fragments. And raster encoding and compression obviously provide for compression of font rasters.

One other compression technique will be mentioned here. It consists of the use in the master of frame variables to store items used frequently in a FontDescription.

For example, the master can store the identifier names of the heavily used character metrics, such as *escapementX*, each in a frame variable using FSET, and then use FGET to retrieve the name and insert it at each appropriate place in the FontDescription. That is, having once executed <*escapementX* 1 FSET>, it is thereafter generally possible to execute <1 FGET>, which encodes in 3 bytes, in place of <*escapementX*>, which takes 13.

For another example, the master can store in frame variables the portions of character mask operators which are repeated frequently within the font or within a character. For example, if certain complex curves or shapes appear frequently within the font, the master can create frame variables containing composed operators to define those curves or shapes, and reference them whenever they are to appear in the FontDescription. Notice that this compression technique for character masks would tend to compress the font master and sometimes but not always the resulting Font; whereas the composite character techniques described earlier would tend to compress both the font master and the resulting Font.

Also notice that this entire technique is somewhat limited by the limited number of frame variables.

4.6 Media and transport and application protocols

This Font Interchange Standard specifies a data format for a font or metric master. It does not, however, specify the media or the transport or application protocols to be used to transport the master or specify its application. Those protocols are separately defined, and are not generally peculiar to fonts.

For example, the Filing Protocol is an application protocol that might be used to move a font master to or from a remote file system. Or the Printing Protocol might be used to request and retrieve a font metric master from a printer. In any case, the application protocol might proceed via Ethernet, or phone wire, or magnetic tape, or floppy disk, using appropriate intermediate transport protocols. Again, these protocols are defined separately from this Font Interchange Standard, and are not generally peculiar to fonts.

In general, whatever media and transport protocols are used for transport of an Interpress master to printers for printing would likely be used also for transport of font masters. The application protocols, however, would likely differ.

A. REFERENCES

International Standards Organization. *7-Bit Coded Character Set for Information Processing Interchange*. ISO 646-1983 (E).

Defines a limited, 7-bit character code mapping.

Xerox Corporation. *Character Code Standard*. Xerox System Integration Standard. Stamford, Connecticut; December, 1985. Version XC1-2-2-0; XNSS 058512.

Defines a comprehensive 16-bit character code mapping, and defines a string encoding format particularly useful for representing strings of character codes.

Xerox Corporation. *Interpress Electronic Printing Standard*. Xerox System Integration Standard. Stamford, Connecticut; January 1986. Version 3.0; XNSS 048601.

Defines and explains the form and meaning of an Interpress master.

Xerox Corporation. *Print Service Integration Standard*. Xerox System Integration Standard. Stamford, Connecticut; December 1985. Version 2.0; XNSS 198512.

Describes various aspects of font handling and font naming in the Xerox environment.

Xerox Corporation. *Raster Encoding Standard*. Xerox System Integration Standard. Stamford, Connecticut; June 1985. Version 1.0; XNSS 178506.

Describes various aspects of the handling of raster images in the Xerox environment.

B. EXAMPLE FONT AND METRIC MASTERS

This appendix contains an example of a font master and of a font metric master.

B.1 Example font master

Following is an example of a font master for two fonts, with only a few characters in each font. The character masks include examples of masks defined by pixel arrays, by strokes, by filled straight-line outlines, and by filled curved outlines; the example character mask shapes are intentionally crude in order to keep the examples simple. For purposes of actual interchange, this example master would be encoded according to the encoding rules given in the Interpress standard; in particular, the pixel array sample data would likely be encoded using one of the compressed pixel vector sequences.

```
--begin example font master--
{                                          --begin instructions body--
  set FIS publishingFonts 2 MAKEVEC
  }                                        --end instructions body--
BEGIN                                      --begin block--
{                                          --begin preamble body--
  --begin elements of a FontDescription vector--
    format FontDescription
    formatVersion 5
    name Xerox XC1-2-2 ExampleTypeFace 3 MAKEVEC
    pixelArrayTransformationUses 1/10 SCALE 1 MAKEVEC
    substituteIndex 61642                  --61642=360|312₈=substitute character--
    transformation 1/30 SCALE
    fontMetrics typefaceName "Example Type Face" 2 MAKEVEC
    characterMetrics
      32 escapementX 10 amplified 1 correction 1 6 MAKEVEC --32=40₈=space--
      45 escapementX 18 2 MAKEVEC          --45=55₈=minus sign--
      70 escapementX 21 2 MAKEVEC          --70=106₈=F--
      111 escapementX 18 2 MAKEVEC         --111=157₈=letter o (lower case letter O)--
      61642 escapementX 16 2 MAKEVEC       --substitute character--
      10 MAKEVEC                           --make characterMetrics vector--
    characterMasks
      45 {                           --minus sign, begin character mask operator body--
        3 15 ISET 0 16 ISET              --set strokeWidth to 3, strokeEnd to 0 (square)--
        3 10 15 10 MASKVECTOR            --horizontal stroke--
        } MAKESIMPLECO                   --make minus sign character mask operator--
```

```
    70 {                                    --F, begin character mask operator body--
      5 7 1 1 1                             --xPixels=5, yPixels=7--
      3 SCALE 3 0 TRANSLATE CONCAT          --pixel array transformation--
        1 1 1 1 1 1 1                   --pixel array samples, scan from bottom to top--
        0 0 0 1 0 0 1                       --and then from left to right--
        0 0 0 1 0 0 1
        0 0 0 1 0 0 1
        0 0 0 0 0 0 1
        35 MAKEVEC                          --make pixel array samples vector--
      MAKEPIXELARRAY MASKPIXEL              --make and mask pixel array--
      } MAKESIMPLECO                        --make F character mask operator--
    111 {                                 --letter o, begin character mask operator body--
      9 0 MOVETO 9 15 9 0 ARCTO             --make circular trajectory for outside of o--
      9 2 MOVETO 9 13 9 2 ARCTO             --make circular trajectory for inside of o--
      2 MAKEOUTLINEODD MASKFILL             --make and mask-fill outline--
      } MAKESIMPLECO                        --make letter o character mask operator--
    61642 {                                 --substitute character--
      3 0 10 21 MASKRECTANGLE} MAKESIMPLECO --filled mask--
    8 MAKEVEC                               --make characterMasks vector--
  18 MAKEVEC  --make FontDescription for Xerox/XC1-2-2/ExampleTypeFace--
--begin elements of another, even shorter FontDescription vector--
  format FontDescription
  formatVersion 5
  name Xerox XC1-2-2 AnotherExampleFace 3 MAKEVEC
  pixelArrayTransformationUses 0 SCALE 1 MAKEVEC
  transformation 1/20 SCALE
  characterMetrics
    32 escapementX 7 amplified 1 correction 1 6 MAKEVEC  --32=40₈=space--
    2 MAKEVEC                               --make characterMetrics vector--
  characterMasks 0 MAKEVEC                  --make characterMasks vector--
  14 MAKEVEC  --make FontDescription for Xerox/XC1-2-2/AnotherExampleFace--
} END                           --end of preamble, end of block, and end of font master--
```

B.2 Example metric master

Following is an example of a font metric master for two fonts, corresponding to the preceding font master example. The easy transformations given are likely values for the preceding example fonts if the fonts were used on a printer with a resolution of 300 spots per inch and normal scan orientation. As with the font master, for purposes of actual interchange this example font metric master would be encoded according to the encoding rules given in the Interpress standard.

```
--begin example font metric master--
{                                           --begin instructions body--
  set FIS simpleFontMetrics 2 MAKEVEC
  }                                         --end instructions body--
BEGIN                                       --begin block--
{                                           --begin preamble body--
  --begin elements of a FontDescription vector--
    format FontDescription
    formatVersion 5
```

```
   name Xerox XC1-2-2 ExampleTypeFace 3 MAKEVEC
   easy 254/300000 SCALE 1 MAKEVEC     --10-pixel body size/300 pixels-per-inch--
   substituteIndex 61642               --61642=360|312₈=substitute character--
   transformation 1/30 SCALE
   fontMetrics typefaceName "Example Type Face" 2 MAKEVEC
   characterMetrics
      32 escapementX 10 amplified 1 correction 1 6 MAKEVEC     --32=40₈=space--
      45 escapementX 18 2 MAKEVEC      --45=55₈=minus sign--
      70 escapementX 21 2 MAKEVEC      --70=106₈=F--
      111 escapementX 18 2 MAKEVEC     --111=157₈=letter o (lower case letter O)--
      61642 escapementX 16 2 MAKEVEC --substitute character--
      10 MAKEVEC                       --make characterMetrics vector--
   16 MAKEVEC          --make FontDescription for Xerox/XC1-2-2/ExampleTypeFace--
--begin elements of another FontDescription vector--
   format FontDescription
   formatVersion 5
   name Xerox XC1-2-2 AnotherExampleFace 3 MAKEVEC
   easy 0 SCALE 1 MAKEVEC
   transformation 1/20 SCALE
   characterMetrics
      32 escapementX 7 amplified 1 correction 1 6 MAKEVEC  --32=40₈=space--
      2 MAKEVEC                        --make characterMetrics vector--
   12 MAKEVEC  --make FontDescription for Xerox/XC1-2-2/AnotherExampleFace--
} END                  --end of preamble, end of block, and end of font metric master--
```

XEROX

Font Interchange Standard

Xerox System Integration Standard

XNSS 238512

Xerox Corporation
Stamford, Connecticut 06904

Printed in U.S.A.

610P50677