Chapter 10: Downloading Fonts

Contents of this Chapter

This chapter describes the following PCL commands:

10.1 Introduction to Soft Fonts

Chapter 10 describes the first part of font downloading: font definition. Chapter 11 describes the second part of font downloading: character definition.

Font Downloading

Downloading a font to the printer consists of the following steps:

1. Define a Font ID for the font and download the font definition.
2. Define the Character Codes and character definitions for each character.

The definition of a font with *n* characters looks like the following:

Font ID
Font Definition
Character $Code_1$
Character $Definition_1$
Character $Code_2$
Character $Definition_2$
...
Character $Code_n$
Character $Definition_n$

Definitions

Font ID: A unique identification number that is designated prior to downloading a font definition. Existing fonts with the same ID are deleted at download.

Font Descriptor: A block of data that describes font design characteristics, such as height, width, style, typeface, and symbol set. The font definition (see below) includes the font descriptor, which always comes first. The first word of the font descriptor defines its size — which does not include any subsequent font definition data segments.

Font Definition: Includes the font descriptor as well as additional data segments such as the Global Intellifont Segment, the Global TrueType Segment, the Copyright, the Application Support Segment, etc. The total size of the font definition — including the font descriptor — is given by the # in *Esc)s#W*, which introduces a font definition.

Character Code: A unique identification number that is designated prior to downloading a character. Characters with the same code in the font being downloaded are deleted at download. (See Chapter 11)

Character Definition: Contatins information about an individual character, such as position and size. The character definition includes data containing either the character's bitmap image or scalable contour. (See Chapter 11.)

Temporary and Permanent Fonts

Downloaded fonts are by default *temporary*, unless designated *permanent*. Temporary fonts are erased by *EscE*; permanent fonts are not. Fonts not used by other jobs should be designated temporary.

Unbound Fonts

Scalable fonts can be downloaded without symbol set affiliation. A ***bound*** font is restricted to a single symbol set. An ***unbound*** font has a larger number of symbols that can be used for multiple symbol sets. Unbound fonts and symbol set downloading are described in Chapter 12.

Bitmap and Scalable Fonts

Both bitmap and scalable fonts use the same downloading process. When the application requests a font, the printer selects a bitmap version of the font if it exists. The font selection process actually controls the scaling process, eliminating the need for a separate font scaling command: the Font Height command (*Esc(s#V*) is the operator for proportional scalable fonts, and the Font Pitch command (*Esc(s#H*) is the operator for fixed-pitch scalable fonts.

10.2 The Font ID

Before sending font data, the font must first be assigned an identification number so the font can be referenced by subsequent PCL commands.

Font ID *Esc * c # d/D*

Specifies an identifying state variable for use in subsequent font management.

Value(#) = ID number
Default = 0
Range = 0 to 2^{32} -1

A font already having this ID number is deleted when the font definition is received, even if the new font is rejected because of memory constraints or invalid data fields.

This ID is used as the value field of the *Esc(#X* and the *Esc)#X* soft font selection commands.

EXAMPLE

Assume that *Esc*c1D* sets the current Font ID to 1. If this command is followed by a valid font definition (*Esc)s#W*), a font with an ID of 1 is created.

If this command is followed by a Font Management command (*Esc*c#F*), the appropriate action is executed for any font currently associated with an ID of 1.

10.3 The Font Definition

The font definition contains all the information needed to define a font. The first part, the *font descriptor*, defines characteristics common to all the characters of a font (e.g., symbol set type, baseline position, character cell width and height, character orientation, symbol set).

Download Font *Esc) s # W [font definition]*

Downloads a font definition and assigns the font the current font ID.

Value(#) = Number of bytes in the font definition
Default = NA
Range = 0 to 2^{32}-1 (command is ignored and the data discarded if there are invalid fields or insufficient memory)

This command must be sent prior to downloading the characters in the font.

Note that this command downloads the entire font ***definition***, which includes the font ***descriptor***, as well as any additional data segments such as the Global Intellifont Segment, the Global TrueType Segment, the Copyright, the Application Support Segment, etc. The # of this command gives the size of the definition; the first word of the definition gives the size of the descriptor. The descriptor, which is the first part of the definition, defines characteristics common to all the characters of a font.

Some devices may not use a font definition or may ignore some fields; but each field should contain a valid value for printer compatibility. Missing data and "reserved" fields should be set to 0; excess data should be discarded.

Six font definitions are shown on the following pages:

- **Bitmap** — This older definition for bitmap fonts is not recommended for new devices.
- **Resolution-Specified Bitmap** — Bitmap font resolution may be specified in dots-per-inch.
- **Intellifont Bound Scalable** — Intellifont scalable fonts restricted to a single symbol set.
- **Intellifont Unbound Scalable** — Intellifont scalable fonts not bound to a single symbol set.
- **TrueType Scalable** — All TrueType scalable fonts, bound and unbound, except 2-byte fonts.
- **Universal** — All scalable and bitmap fonts, including 2-byte fonts. (Recommended)
- **Older DeskJet** — Used in the DeskJet 5xx family (except 540).

Bitmap Font Definition

The bitmap font definition is shown below.

Byte	15 (MSB) 8	7 0 (LSB)	Byte
0	Font Descriptor Size (64)		1
2	Descriptor Format (0)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol Set		15
16	Pitch (Default CMI)		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	First Code		37
38	Last Code		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Copyright (optional)		65
n	...		n+1

DEVICE NOTE: For the bitmap font definition, LJs support a descriptor size of less than 64; any fields not read in are set to 0 except Underline Position, which is set to 5.

Resolution-Specified Bitmap Font Definition

The font definition below is the same as the previous one, except that it allows the specification of resolution. Grayed fields show the differences.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥68)		1
2	Format (20)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol Set		15
16	Pitch (Default CMI)		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	First Code		37
38	Last Code		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	X Resolution		65
66	Y Resolution		67
68	Copyright (optional)		69
n	...		n+1

Intellifont Bound Scalable

The Intellifont bound scalable font definition is shown below. Fields that differ from the bitmap definition are grayed.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥80)		1
2	Descriptor Format (10)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Master Design Pitch (default CMI)		17
18	Height		19
20	x Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Reserved (0)		33
34	Reserved		35
36	First Code		37
38	Last Code		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Scale Factor		65
66	Master X Resolution		67
68	Master Y Resolution		69
70	Master Underline Position		71
72	Master Underline Height		73
74	OR Threshold		75
76	Global Italic Angle		77
78	Global Intellifont Data Size		79
80	Global Intellifont Data		81
82	Copyright (optional)		83
n	. . .		n+1
	Reserved (0)	Checksum	

Intellifont Unbound Scalable

The Intellifont unbound scalable font definition is shown below. Fields differing from the bitmap font definition are grayed.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥88)		1
2	Format (11)	Symbol Set Type (10)	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Pitch		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	Reserved (0)		37
38	Number of Contours (characters)		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Scale Factor		65
66	Master X Resolution		67
68	Master Y Resolution		69
70	Master Underline Position		71
72	Master Underline Height		73
74	OR Threshold		75
76	Global Italic Angle		77
78-84	Character Complement		79-85
DescSize-2	Global Intellifont Data Size		DescSize-1
DescSize	Global Intellifont Data		DescSize+1
n	Copyright (optional) ...		n + 1
	Reserved (0)	Checksum	

TrueType Scalable

The TrueType scalable font definition is shown below. Fields differing from the bitmap font definition are grayed.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥72)		1
2	Descriptor Format (15)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol Set		15
16	Pitch		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	First Code		37
38	Last Code/Number of Characters		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Scale Factor		65
66	Master Underline Position		67
68	Master Underline Thickness		69
70	Font Scaling Technology	Variety	71
72	{additional data may be inserted here}		Desc. Size-1

Desc. Size	Segmented Font Data		
	...		...
			# - 3
# - 2	Reserved (0)	Checksum	# - 1

Universal

The Format 16 font definition is similar to the Format 15 definition. However, Format 16 allows the downloading of two-byte bound fonts. Format 16 goes beyond Format 15 in that it:

- Supports Symbol Set Type 3 (bound, 16-bit symbol set) as well as unbound symbol sets.
- Supports bitmap and scalable font scaling technologies (Format 15 only supported scalable).
- Increases the Segmented Font Data field from 16 to 32 bits.
- Supports additional data segments.
- Supports dual-pitch spacing.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥72)		1
2	Descriptor Format (16)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol Set		15
16	Pitch (Nominal)		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	First Code		37
38	Last Code/Number of Characters		39
40	Pitch Extended (Nominal)	Height Extended	41
42	Cap Height		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Scale Factor		65
66	Master Underline Position		67
68	Master Underline Thickness		69
70	Font Scaling Technology	Variety	71
72	{additional data may be inserted here}		Desc. Size-1

Desc. Size	Segmented Font Data		
	...		...
			# - 3
# - 2	Reserved (0)	Checksum	# - 1

The following notation is used to define data types in the font definition:

(BOOL)	Boolean (0,1)	(SINT16)	Signed Integer (-32768 .. 32767)
(UBYTE)	Unsigned Byte (0 .. 255)	(UINT32)	Unsigned Long Integer (0 .. (2^{32}-1))
(SBYTE)	Signed Byte (-128 .. 127)	(SINT32)	Signed Long Integer (-2^{31} .. (2^{31}-1))
(UINT16)	Unsigned Integer (0 .. 65535)	(ASCxx)	ASCII String (array (0 .. (xx-1)) of characters

Font Descriptor Size (UINT)

The number of bytes in the font descriptor (this is not the font *definition* size, which is given by the escape sequence value field). The font is invalid if the size is less than the minimum required; for example, a Format 16 download is ignored if the descriptor size is less than 72.

Descriptor Format (UBYTE)

Value	Format
0	Standard Bitmap
5	DeskJet Bitmap
6	PaintJet Bitmap
7	PaintJet XL Bitmap
9	DeskJet Plus Bitmap
10	Bound Intellifont Scalable
11	Unbound Intellifont Scalable
12	DeskJet 500 Bitmap
15	TrueType Scalable
16	Universal
20	Resolution-Specified Bitmap

Unrecognized values invalidate font creation.

DEVICE NOTE: DJ5xx's use a value of 9 for landscape fonts or fonts larger than 18 points with positive escapements, and a value of 12 for negative escapements. All other DJ5xx fonts use a value of 5.

Symbol Set Type (UBYTE)

Describes the font's relationship to symbol sets.

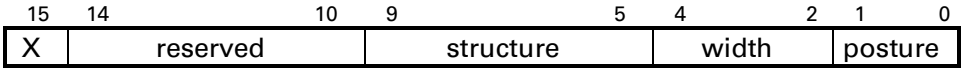
Value	Symbol Set Organization
0	Bound font, 7-bit (96 characters) — Character codes 32-127 [decimal] are printable.*
1	Bound, 8-bit (192 characters) — Character codes 32-127 and160-255 printable.*
2	Bound, 8-bit (256 characters) — All codes are printable except 0, 7-15, and 27.*
3	Bound, 16-bit (65535 characters) — All are printable except 0, 7-15, 27, 65279, 65534, 65535
10	Unbound — Character codes correspond to MSL numbers (Intellifont).
11	Unbound — Character codes correspond to Unicode numbers (TrueType).

***** Access to unprintable codes with a defined character requires Transparent Data Mode (*Esc&p#X*).

Style MSB (UINT16)

The style MSB combines with the style LSB to make the style word, which is calculated from the partial sums for posture, width, and structure. The binary structure of the style word is:

Style Word = Posture + (4 x Width) + (32 x Structure)



Value(#)= **Posture** (style word partial sum)

0	-	Upright
1	-	Italic
2	-	Alternate Italic
3	-	Reserved

= **Width** (style word partial sum multiplied by 4)

0	-	Normal
1	-	Condensed
2	-	Compressed or extra condensed
3	-	Extra compressed
4	-	Ultra compressed
5	-	Reserved
6	-	Extended or expanded
7	-	Extra extended or extra expanded

= **Structure** (style word partial sum multiplied by 32)

0	-	Solid
1	-	Outline
2	-	Inline
3	-	Contour, Edge effects
4	-	Solid with shadow
5	-	Outline with shadow
6	-	Inline with shadow
7	-	Contour with shadow
8-11	-	Patterned (complex patterns, subjective to typeface)
12-15	-	Patterned with shadow
16	-	Inverse
17	-	Inverse in open border
18-30	-	Reserved
31	-	Unknown structure

DEVICE NOTE: DeskJets prior to DJ500 and LaserJets prior to LJII cannot "see" bits 8-15.

The reserved bits (10 to 14) should be set to 0.

Baseline Position (UINT16)

Bitmap Font - Specifies the distance from the baseline (an imaginary dot row on which the characters stand) to the top of the cell. The measurement is in font resolution dots as defined in the Resolution Field of a Format 20 font definition (default=300 dpi). Since the baseline position must be contained within the cell, the legal value for the baseline position may range from zero to cell height minus one.

Intellifont - Specifies the location of the baseline (as a Y coordinate) within the design window coordinate system. The typically observed value is 5380.

TrueType - Set to 0.

DEVICE NOTE: Post-LJIIs ignore this field. DJ5xx, which has with a cell height of 50, uses 1-49.

Cell Width (UINT16)

Specifies the width from the leftmost extent of any character in the font to the rightmost extent of any character in the font. The legal range is 1 to 65535.

Bitmap Font - Specified in PCL coordinate system dots.

Intellifont - Specified in design window units (defined in the Scale Factor field).

TrueType - Suggested value is X_{MAX} - X_{MIN} (as obtained from the head table of the TrueType font).

DEVICE NOTE: Post LJIID's and DJ5xx's except DJ540 ignore this value.

Cell Height (UINT16)

Specifies the distance from the lowest descent of any character in the font to the highest ascent of any character in the font. The legal range is 1 to 65535.

Bitmap Font - Specified in PCL coordinate system dots.

Intellifont - Specified in design window units (defined in Scale Factor field).

TrueType - Suggested value is Y_{MAX} - Y_{MIN} (as obtained from the head table of the TrueType font).

DEVICE NOTE: Post LJIID's and DJ5xx's except DJ540 ignore this value.

Orientation (UBYTE)

Specifies font orientation. All font characters must have the same orientation as those specified in the font descriptor; otherwise they are discarded as they are downloaded.

0 = portrait (0 degrees)
1 = landscape (90 degrees counterclockwise)
2 = reverse portrait (180 degrees counterclockwise)
3 = reverse landscape (270 degrees counterclockwise)

Intellifont - Set to 0.

TrueType - Set to 0.

DEVICE NOTE: DJ5xx supports supports 0 and 1. Post-LJII printers rotate the fonts to match the media's orientation.

Spacing (UBYTE)

Value	Format
0	Fixed
1	Proportional
2	Dual-fixed

DEVICE NOTE: DJ5xx, except DJ540, treats values other than 0 or 1 as 1 and requires landscape fonts to have fixed spacing.

DEVICE NOTE: LJ4V and LJ4PJ do not support the dual fixed pitch value.

Symbol Set (UINT16)

Bound Font

Specifies the symbol set characteristic of the font.

The value for this field is derived from the symbol set identification number (ID) used by *Esc(ID* in font selection (see Chapter 9 for HP-supported symbol sets and IDs). The number portion (#) and the ASCII value of the letter portion (L) of the ID are used to obtain the symbol set descriptor field value:

Symbol Set Descriptor Field = (# × 32) + (L - 64)

For example, assume the symbol set is US ASCII ISO-6. The symbol set table in Chapter 9 identifies US ASCII as "0U". Since # = 0 and U = 85, the field value is 21:

Symbol Set Descriptor Field = (0 × 32) + (85 - 64) = 21

DEVICE NOTE: LJs may use font descriptor symbol set values 0 to 1023. Values 1024 to 2047 are available for use by third-party font vendors.

Unbound Font

This field should be set to 56 (1X) for unbound fonts.

Pitch (UINT16)

Bitmap Font - Specifies the pitch of the font in quarter-dot units (i.e., four quarter-dot units equal one dot; also known as radix dots). It combines with Pitch Extended to specify the pitch of the font in 1/1024 dots. Pitch defines the default CMI for the font.

For example, at 300 dpi (1200 quarter-dots/inch), a 17 cpi font has a pitch field of 70 and a non-zero pitch extended field. (1inch / 17char) x (300 dots / inch) x (4 radix dots / dot) = 70.588 radix dots/char. The remainder 0.588 is converted back to dots and then to 1/1024 dots: (0.588 radix dots / 4 radix dots per dot) x (1024 units / dot) = 150 units. Pitch Extended is set to 150 1/1024 units.

For proportional fonts, the width "printed" for a control code space is determined by the pitch value unless CMI has been changed.

Scalable Font - Contains the master design width of the font in design window units (defined in the Scale Factor field).

Dual-fixed Spacing - This field contains the pitch for the nominal space, which is the largest width for a space character in a dual-fixed pitch font.

DEVICE NOTE: LJ+ supports 2-1260; LJII supports 0-16800, with greater values clamped.

Height (UINT16)

Bitmap Font - Specifies font height in quarter-dots. The value, converted to points (1/72 inch), is the font's height characteristic. Height and Height Extended specify the font's design height in 1/1024 dots. Thus, a 10 pt font at 300 dpi has a height field of 166 quarter dots (1200 quarter dots/inch, 1/72 inch/point):

(10 point) x (1 inch / 72 point) x (300 dots/inch) x (4 quarter-dots/dot) = 166.667

Intellifont - Specifies the font's master design height in 1/8 points. The typical value is 2000.

TrueType - Set to 0.

DEVICE NOTE: LJ+ and LJII support 0 to 10922, with greater values clamped

xHeight (UINT16)

Bitmap Font - Specifies the height of the lower case "x" in quarter dots.

Scalable Font - Specifies the distance from the baseline to the lower case "x" height in design window units (as defined in the Scale Factor field).

Width Type (SBYTE)

Specifies the proportionate width of characters in the font.

Value	Appearance Width
-5	Ultra Compressed
-4	Extra Compressed
-3	Compressed, Extra Condensed
-2	Condensed
-1	Semi-Condensed
0	Normal
1	Semi-Expanded
2	Expanded
3	Extra Expanded

Style LSB (UBYTE)

The least significant byte of the Style word. Refer to the Style MSB field.

Stroke Weight (SBYTE)

Specifies the thickness of the font characters. The standard stroke weight is 0 for a medium font, 3 for a bold font, and -3 for a light font.

Value	Stroke Weight
-7	Ultra thin
-6	Extra thin
-5	Thin
-4	Extra light
-3	Light
-2	Demi-light
-1	Semi-light
0	Medium, Book, or Text
1	Semi-bold
2	Demi-bold
3	Bold
4	Extra bold
5	Black
6	Extra black
7	Ultra black

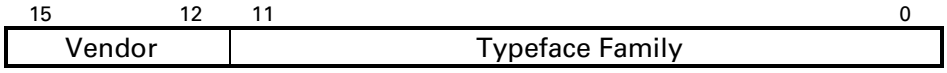
Default = 0
Range = -7 to 7 (less than -7 maps to -7; greater than 7 maps to 7)

Typeface [LSB/MSB] (UBYTE individually, UINT16 together)

Specifies the HP typeface number. Three versions of this field are used: the obsolete single-byte version, the DeskJet 500 / LaserJet III version, and the current version.

CURRENT VERSION

The current version is shown below. Printers may treat the typeface number as a single value and ignore a request in which a match cannot be made (for selection purposes, the font select table is updated). The procedure for allocating typeface numbers for the font products of various vendors, however, will consider the typeface number to be composed of two distinct fields: a vendor field consisting of the four most significant bits and a typeface family field consisting of the the 12 least significant bits. If a match with the full 16-bit word cannot be obtained, bits 12-15 are masked and a match is attempted with bits 0-11.



Vendor Number (bits 12-15) — This HP-assigned value is between 0 and 15.

Value	Vendor
0	Reserved
1	Agfa Division, Miles Inc.
2	Bitstream Inc.
3	Linotype Company
4	The Monotype Corporation plc
5	Adobe Systems, Inc
6-15	Reserved

Typeface Family Number (bits 0-11) — This value is between 0 and 4095 and is calculated according to the following formula using the typeface base values listed in Chapter 9:

Typeface Family Number = Typeface Base Value + (Vendor Value x 4096)

For example, the HP typeface number for Agfa Dom Casual typeface is 4157 (typeface base value = 61 and vendor value =1) or 61 + (1 x 4096).

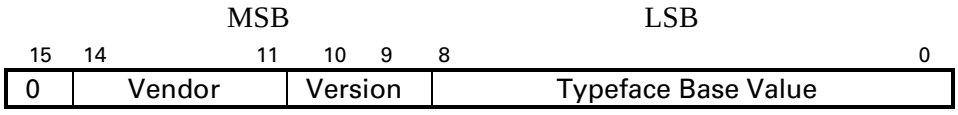
SINGLE-BYTE VERSION

Pre-DeskJet 500s and pre-LaserJet IIIs used only the least significant byte (LSB) of the typeface word and ignored the upper byte (MSB).

LASERJET III / DESKJET 500 VERSION

The typeface word includes a 4-bit field for the vendor number, a 2-bit field for the version number, and a 9-bit field for the typeface number. The most significant bit of the most significant byte is zero.

Typeface Family = Typeface Base Value + (Version x 512) + (Vendor x 2048)



Typeface Base Value

0	Line Printer or Line Draw
3	Courier
4	Helvetica
5	Times Roman
6	Letter Gothic
... (See Chapter 9 for a complete list)	

Version (typeface word partial sum multiplied by 512)

0	1st version
1	2nd version
2	3rd version
3	4th version

Vendor (typeface word partial sum multiplied by 2048)

0	Reserved for generic typeface selection
1	Reserved for HP use only
2	Agfa Division, Miles Inc.
4	Bitstream Inc.
6	Linotype Company
8	The Monotype Corporation plc
10	Adobe Systems, Inc.
3, 5, 7, 9, 11-15	reserved
Default	3
Range	0 to 65535

Vendor Number (bits 11 to 14) - This HP-assigned value is between 0 and 15.

Vendor Version (bits 9, 10) - This value is between 0 and 3. It will change when the vendor changes the width of a font or adds new characters to a font. A vendor code of 0 is reserved for generic typeface selection so that older one-byte typeface values can still be used in the generic typeface selection process.

Typeface Base Value (bits 0 to 8) - This value is between 0 and 511. Some of these values include appearance width and structure information (i.e., Helvetica Compressed and Helvetica Outline, etc.). See Chapter 9 for a list of valid typeface families and their numbers.

A typeface family value in which both Vendor and Version numbers are 0 is reserved for generic typeface selection. That is, for typeface family values less than 512, the printer exactly matches the LSB typeface base value field. For typeface values greater than or equal to 512, the full 16-bit typeface word is used.

For example, the HP typeface number for Agfa's Dom Casual typeface is 4157 (vendor value = 2, version value = 0, and typeface value = 61). That is, 61 + (0 × 512) + (2 × 2048) = 4157.

Serif Style (UBYTE)

Specifies one of the following defined serif styles. Values 0 to 63 (the lower six bits) are ignored for bitmap fonts; but the upper two bits (6, 7) are used for scalable fonts to determine the serif style of the typeface insensitive characters to complement the font.

Value	Serif Style
0	Sans Serif Square
1	Sans Serif Round
2	Serif Line
3	Serif Triangle
4	Serif Swath
5	Serif Block
6	Serif Bracket
7	Rounded Bracket
8	Flair Serif, Modified Sans
9	Script Nonconnecting
10	Script Joining
11	Script Calligraphic
12	Script Broken Letter
13-63	Reserved
Values for bits 6 and 7	
64	Sans Serif
128	Serif
192	Reserved

DEVICE NOTE: DJ5xx, LJIII, and earlier printers ignore this field. LJIIIP and LJ4 ignore 0-63, 65-127, 129-191, and 193-255 for bitmap fonts. Scalable fonts use the two most significant bits of this field to determine the serif style of the typeface-insensitive characters to complement the font.

Quality (UBYTE)

Specifies the quality or density of the font.

Value	Quality
0	Data Processing (Draft)
1	Near Letter Quality
2	Letter Quality

DEVICE NOTE: LaserJets ignore this field.

Placement (SBYTE)

Specifies the position of character patterns relative to the baseline.

Bitmap Font - The placement values for bitmap fonts are listed below.

Value	Position
1	Superior (superscript)
0	Normal
-1	Inferior (subscript)

Scalable Font - Set to 0.

DEVICE NOTE: All DJ5xx fonts are treated as normal. LaserJets ignore this field.

Underline Position (SBYTE)

Bitmap Font - Specifies the distance from the baseline to the top dot row of the underline in dots. Zero specifies an underline position at the baseline. A positive value specifies an underline position above the baseline. A negative value specifies an underline position below the baseline.

Scalable Font - Set to 0. Underline Distance is ignored. The Master Underline Position field identifies this information for scalable fonts.

DEVICE NOTE: In DJ5xx, single and double underlines occupy the character cell's bottom 12 dots.

Underline Thickness (UBYTE)

Specifies the thickness of the underline in dots for a bitmap font.

Bitmap Font - Specified in dots. A bitmap font prints 3-dot (1/100") thick underlines at 300 dpi and 6-dot thick underlines at 600 dpi.

Scalable Font - Should be ignored and set to 0. The Master Underline Height provides this information.

DEVICE NOTE: All LaserJets after LJIII print 1/100" thick underlines.

Text Height (UINT16)

Specifies the font's optimum inter-line spacing. This value is typically 120% of the height of the font.

Bitmap Font - Specified in quarter-dot units (radix dots).

Scalable Font - Specified in design window units (defined in the Scale Factor field).

DEVICE NOTE: DJ5xx ignores this field.

Text Width (UINT16)

Specifies the font's average lowercase character width (which can be weighted on the basis of relative frequency).

Bitmap Font - Specified in quarter-dots (radix dots).

Scalable Font - Specified in design window units (defined in the Scale Factor field).

DEVICE NOTE: DJ5xx ignores this field.

First Code (UINT16)

Specifies the character code of the first printable character in the font. The space character may be printable, and will print an image if one is defined; otherwise, a space control code is executed.

First Code must be less than or equal to Last Code. For a bound two-byte font, this value may be any value between 0 and 65535.

DEVICE NOTE: LaserJets and DeskJets 540, 660, and 850 ignore this field and use the Symbol Set Type field to determine first and last codes, as follows:

Symbol Set Type	First Code/Last Code
0	32/127
1	32/127 - 160/255
2	0/255
3	0/65535
10	Set to 0 (for unbound font)
11	Set to 0 (for unbound font)

Last Code / Number of Characters (UINT16)

Specifies the last downloadable character code in the font. This value may be greater than the last code of the symbol set as implied by the symbol set type because there may be components of compound characters that are not part of the symbol set, but must be downloaded. The First Code must be less than or equal to Last Code. For a bound two-byte font, this value may be between 0 and 65535.

DEVICE NOTE: LaserJets and DJ540, 660, and 850 ignore this field and use the Symbol Set Type field to determine first and last codes, as described under First Code.

Unbound Font - For an unbound font (symbol set type 10 or 11), this field specifies the maximum number of characters that can be downloaded into the font.

Pitch Extended (UBYTE)

Bitmap Font - This is an addition to the Pitch field which extends pitch an extra eight bits to allow 10 bits of fractional dots. The value of this field is in font design units (as defined in the Scale Factor field). For example, a 17 cpi pitch font designed at 300 dpi has a Pitch field of 70 (17.5 dots or 17.1429 cpi) and a Pitch Extended field of 150 (0.1465 dots additional, which adds to 17.6465 dots, or 17.0005 cpi). An example of calculating the Pitch and Pitch Extended fields is provided in the Pitch field description.

Scalable Font - This field is set to zero.

Height Extended (UBYTE)

Bitmap Font - This is an addition to the Height field which extends the height an extra eight bits to allow 10 bits of fractional dots. The value of this field is in font design units (as defined in the Scale Factor field). For example, a 10 point font designed at 300 dpi has a height of 166 (41.5 dots, or 9.96 points) and a Height Extended field of 170 (0.1660 dots additional, which adds to 9.9998 points). This field is similar to the Pitch Extended field (refer to the Pitch field example).

Scalable Font - This field is ignored and should be set to zero.

Cap Height (UINT)

Cap height is a percentage of the Em of a font and is used to calculate the distance from the capline (top of an unaccented, upper-case letter, e.g., "H") to the baseline. *Em* is a measure in decipoints of the height of a font; e.g., the em of a 10-point font is 100 decipoints.

Bitmap Font - Fonts containing a 0 in this field are assumed to have a cap height percentage of 70.87% of em. The Cap Height data is represented as the product of the cap height percentage and the maximum unsigned integer:

0.7087 x 65535 = 46445

For nonzero values the Cap Height percentage is calculated as follows:

% = (Cap Height Data / 65535) x 100

Scalable Font - Contains the cap height in design units (as defined in the Scale Factor field).

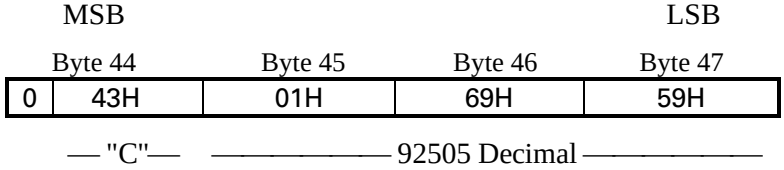
Font Number (UINT32)

Bitmap Font - Should be ignored and set to 0.

Scalable Font - This field uses four bytes (44-47). The lower three bytes (45-47) contain the vendor font number in hexadecimal. The most significant bit of the most significant byte (44) is a flag indicating whether the font is in its native (0) format or has been converted (1); the lower seven bits contain the ASCII decimal value for the first initial of the vendor's name. The following initials have been assigned:

Initial	Hex Value	Vendor Name
A	41	Adobe Systems
B	42	Bitstream Inc.
C	43	Agfa Division, Miles Inc.
H	48	Bigelow & Holmes
L	4C	Linotype Company
M	4D	Monotype Corporation plc

For example, to identify the Font Number for a CG Times Bold Italic native format font:



Font Name (ASC16)

This is a 16-byte ASCII character field in which the user may assign a font name. The font name is used in the Typeface list (or font list printout) under *Name* or *Typeface* if the printer does not have a name string assigned to the typeface family code in its font selection table.

DEVICE NOTE: DJ5xx prints the name as part of the printer self-test. LJIII and later LJs use the font name in the Font List printout under "Name" or "Typeface", or use this field if the typeface family code and treatment is unknown. This field is also used by several applications to provide user-friendly identification.

Scale Factor (UINT16)

This field indicates the number of design units per Em and is the unit used for all scalable metrics in the font definition. For information regarding Intellifont, refer to the Scale Factor field of the Attribute Header segment, as described in *Intellifont Scalable Typeface Format*. For information regarding TrueType, refer to the Units Per Em field of the head table, as described in *TrueType Font Files*.

Master X Resolution (UINT16)

This is the pixel resolution in the X scan direction at which the font was designed. For detailed information on X Resolution, refer to the *Intellifont Scalable Typeface Format* document.

X Resolution (UINT16)

In resolution-specified bitmap fonts, this field specifies the resolution of the font in the X dimension in dots per inch.

Master Y Resolution (UINT16)

This is the pixel resolution in the Y scan direction at which the font was designed. For detailed information on Y Resolution, refer to the *Intellifont Scalable Typeface Format* document.

Y Resolution (UINT16)

In resolution-specified bitmap fonts, this field specifies the resolution of the font in the Y dimension in dots per inch.

Master Underline Position (SINT16)

This is the position of the scalable underline with respect to the baseline on the master design grid in design window units as defined in the Scale Factor field. It is used to implement the floating underline command for scalable fonts. For scalable fonts, it replaces the Underline Position field.

Master Underline Thickness (UINT16)

This is the thickness of the underline on the master design grid in dimensional units. as defined in the Scale Factor field. It replaces the 1-byte Underline Height field.

Font Scaling Technology (UBYTE)

Value	Font Scaling Technology
0	Intellifont
1	TrueType
254	Bitmap (defined for Format 16 only)

An undefined value in this byte invalidates the font.

DEVICE NOTE: LJ4V and later support 1 and 254 for Format 16. LJ4PJ supports only 1 for Format 16. All other LJs support value 1 for Format 15.

Variety (UBYTE)

The interpretation of this field depends on the value of the Font Scaling Technology field. For example, if Font Scaling Technology is 0, a Variety of 0 implies that only HQ2 and HQ3 character contour data will be found; and a Variety of 1 implies that HQ4 contour data will be found in addition to HQ2 and HQ3. (As of December 1994, for TrueType fonts, only variety 0 is valid.)

OR Threshold (UINT16)

Formerly called the "Low Resolution Enhancement (LRE) Threshold", this is the pixel size in design units (as defined in the Scale Factor field) above which the missing pixel recovery process is switched on in Intellifont scaling and rasterization.

The size of a pixel in design units increases as point size or device resolution decreases.

Global Italic Angle (SINT16)

This field contains the tangent of the italic angle (relative to the vertical) times 2^{15}. It should be set to zero for upright fonts For detailed information, refer to *Intellifont Scalable Typeface Format*.

Character Complement (Array of UBYTE)

This 8-byte field qualifies the compatibility of a type 10 or 11 unbound font with various symbol sets. Except for the last three bits, each bit is independently interpreted. (Bit 63 refers to the most significant bit of the first byte; bit 0 refers to the least significant bit of the eighth byte.)

In Formats 15 and 16, the data in this field is contained in the "CC" (Character Complement) field in the data segment.

NOTE: There are two symbol indexes used in unbound fonts and symbol set maps. Unbound Intellifonts are ordered in MSL numbers; unbound TrueType fonts are ordered in Unicode numbers. The character collections and character complements are different in Intellifonts than in TrueType fonts. Character complements for each symbol index are listed separately below.

MSL Symbol Index Character Complements

Bit Field	Designated Use
58-63	Reserved for Latin fonts
55-57	Reserved for Cyrillic fonts
52-54	Reserved for Arabic fonts
49-51	Reserved for Greek fonts
47-48	Reserved for Hebrew fonts
46	Thai
3-45	Miscellaneous Uses (South Asian fonts, East Asian fonts, Armenian, Georgian, Amharic, other alphabets, bar codes, OCR, Math, PC Semi-graphics, Dingbats, etc.)
0-2	Symbol Index Indicator

Individually defined bits include:

Bit	Value
63	0 if font is compatible with basic Latin symbol sets (e.g., ISO 8859-1 Latin 1) 1 otherwise.
62	0 if font is compatible with East European symbol sets (e.g., ISO 8859-2 Latin 2); 1 otherwise.
61	0 if font contains Turkish symbol sets (e.g., ISO 8859/9 Latin 5); 1 otherwise.
34	0 if font has access to characters of Math-8, PS Math, and Ventura Math; 1 otherwise.
33	0 if font has access to the semi-graphic characters of PC-8, PC-850, etc.; 1 otherwise.
32	0 if font is compatible with ITC Zapf Dingbats series 100, 200, etc; 1 otherwise.
2,1,0	111 if font is arranged in MSL Symbol Index Order

There are no invalid Character Complement field values. Examples include:

Value (hex)	Meaning (MSL sub-group)
0x0000000000000000	Default complement; font is compatible with every symbol set.
0x7fffffffffffffff	font is only compatible with Basic Latin symbol sets.
0x3fffffffffffffff	font is compatible with standard and East European Latin symbol sets.
0xfffffffeffffffff	font is only compatible with ITC Zapf Dingbat symbol sets.

Unicode Symbol Index Character Complements

Bit Field	Designated Use
32-63	Reserved
28-31	Reserved for symbols found in Latin fonts
26-27	Reserved for accents and publishing symbols
22-25	Reserved for application/environment specific symbols
3-21	Reserved for miscellaneous uses
0-2	Reserved for Symbol Index Indicator

Individually defined bits include:

Bit	Value	
31	0	ASCII (various definitions).
	1	No basic Latin alphabet.
30	0	Latin 1 extensions.
	1	No West European added symbols.
29	0	Latin 2 extensions.
	1	No East European added symbols.
28	0	Latin 5 extensions.
	1	No Turkish and West European added symbols.
27	0	Publishing symbols.
	1	No added publishing symbols.
26	0	Accents (often missing from some sets).
	1	No added accents.
25	0	PCL (Roman-8, Legal, etc.).
	1	Insufficient symbols for the above.
24	0	Macintosh.
	1	Insufficient symbols for MacIntosh text.
23	0	PostScript.
	1	Insufficient symbols for PostScript text.
22	0	Code page.

	1	Insufficient symbols for PC-8, PC-8 D/N.
2,1,0	110	Unicode Symbol Index Indicator.

There are no invalid Character Complement field values. Examples include:

Value (hex)	Meaning (Unicode sub-group)
0xffffffff3ffffffe	Font is compatible with standard West European Latin symbol sets.
0xffffffff5ffffffe	Font is compatible with standard East European Latin symbol sets.

Global Intellifont Data Size (UINT16)

This is size of the Global Intellifont data block. For detailed information, refer to *Intellifont Scalable Typeface Format*.

In the Format 15 and 16 definitions, the data in this field is contained in the Segment Size field data segment.

DEVICE NOTE: LJs accept 0.

Global Intellifont Scalable Data

See the *Intellifont Scalable Typeface Format* for a detailed description.

In the Format 15 and 16 definitions, the data in this field is replaced by the "GI" (Global Intellifont Data) field in the data segment.

Checksum (UBYTE)

This value is computed on bytes 64 through the end of the font definition. The checksum, when added to the sum of byte 64 through the reserved byte, should be 0 when divided by 256 (modulo 256 arithmetic). For example, if the sum of byte 64 through the reserved byte is 10,234, then 10,234 mod 256 is 250. Therefore, the checksum should be 6 (since 250 + 6 = 256 is 0 [mod 256]).

In Formats 15 and 16, the data in this field is located in the data segment.

Copyright (ASCII)

This optional field contains ASCII data and is optional. In Formats 15 and 16, this field is located in the data segment.

Data Segments

The Segmented Font Data section immediately follows the descriptor of a format 15 or 16 font. The segment size for a Format 16 font is 32 bits. Format 15 supports all the following segments except Bitmap Resolution, Character Enhancements, Dual Pitch, Galley Character, Typeface String, Vertical Substitution, and Vertical Rotation Offset.

Each segment contains three parts: a **Segment Identifier, Segment Size** and **Data Segment**. The Segmented Font Data section is terminated by the Null Segment. If no Null Segment is encountered prior to the end of the font definition — as defined in the initial escape sequence — the font is invalidated. Encountering the Null Segment too soon (prior to byte # - 8, as shown below) also invalidates the font.

The figure below shows the general structure of the Segmented Font Data section. (x=Font Descriptor Size; # is the font definition length defined in the escape sequence.)

NOTE: The template below, which has a 32-bit Segment Size field, is the **Format 16 version**. Format 15 has only a 16-bit Segment Size field.

Byte	**15 (MSB) 8**	**7 (LSB) 0**	**Byte**
x	1st seg, Segment Identifier		x+1
x+2 x+4	1st seg, Segment Size		x+3 x+5
x+6 ...	1st seg, Data Segment ...		x+7 ...
x+6+ 1st seg size	2nd segment ...		
...	...		
# - 8	Null Segment Identifier (0xFFFF)		# - 7
# - 6 # - 4	Null Segment Size (0)		# - 5 # - 3
# - 2	Reserved (0)	Checksum	# - 1

Format 16 Data Segment (Format 15 only has a 16-bit Segment Size field)

Segment Identifier (UINT16) — Each entry in the Segmented Font Data section has its own unique identification number.

Value	Mnemonic*	Data Segment
16720	AP	Application Support
16978	BR	Bitmap Resolution (Format 16 only)
17219	CC	Character Complement
17221	CE	Character Enhancements (Format 16 only)
17232	CP	Copyright
17488	DP	Dual Pitch Space Character Code (Format 16 only)
18243	GC	Galley Character (Format 16 only)
18249	GI	Global Intellifont Data
18260	GT	Global TrueType Data
18758	IF	Intellifont Face Data
20545	PA	PANOSE Description
20550	PF	PS-Compatible Font Name
21574	TF	Typeface String (Format 16 only)
22098	VR	Vertical Rotation (Format 16 only)
22100	VT	Vertical Substitute (Format 16 only)
22618	XW	X Windows Font Name
65535		Null Segment

*The mnemonic is obtained when the two bytes of this big-endian word are treated as ASCII characters.

Segment Size (UINT32 for Format 16; UINT16 for Format 15) — For each entry in the segmented font data section, this field indicates the number of bytes in the immediately following data segment. The size for the Null Segment is 0.

AP — Application Support Segment

This segment's definition is reserved. It will replace the "unsanctioned" Application Support Segment that Type Director writes into Format 0 (standard bitmap) and 10 (bound Intellifont) font definitions.

BR — Bitmap Resolution Segment (Format 16 only)

This segment defines the X and Y resolutions of a resolution-specified downloaded bitmap. A bitmap font is invalidated unless it is present. The font is invalidated if the specified resolution is unsupported. (Format 16 only)

Byte	15 (MSB) (LSB) 0	Byte
0	BR (16978)	1
2-4	Segment Size	3-5
6	X Resolution	7
8	Y Resolution	9

X Resolution (UINT16) — Specifies the resolution of the font in the X dimension in dots per inch.

Y Resolution (UINT16) — Specifies the resolution of the font in the Y dimension in dots per inch.

CC — Character Complement Segment

This segment has the same form (i.e., 8 unsigned bytes) and function as the Character Complement of Format 11 (unbound Intellifont) fonts. The Character Complement field should be present with unbound fonts (Symbol Set Types 10 and 11), but is not needed for bound fonts (Symbol Set Types 0, 1, 2, and 3). (Format 16 fonts only).

CE — Character Enhancements Segment (Format 16 only)

Bitmap downloads may have data segments for character enhancements (outline, shadow, pseudo italics and pseudo bolding). CE is also supported for scalable fonts. CE indicates if the downloaded font is allowed to use the printer's character enhancement algorithms.

DEVICE NOTE: LJ4V and 4PJ support CE only in the scalable case.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	CE (17221)		1
2 4	Data Segment Size (8)		3 5
6 8	Style		7 9
10	Stroke Weight		11
12	Sizing		13

* The Data Segment Size field for Font Format 15 is 2 bytes

Style (UINT32) — The types of style the printer may use for the font characters.

Style Word = Posture + Structure

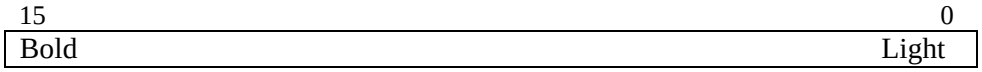
31 12	11 4	3 0
Structure	Reserved	Posture

Bit Positions	
	Posture
1	Italics
0,2,3	Reserved
	Structure
12	Outline
13	Shadow
14-31	Reserved

DEVICE NOTE: DJ540 supports only Italics, Outline, and Shadow styles.

DEVICE NOTE: LJ4V and LJ4PJ support only italics (scalable only).

Stroke Weight (UINT16) — The thickness of the font characters.



Bit Positions	Stroke Weight
0	Reserved
1	Ultra Thin
2	Extra Thin
3	Thin
4	Extra Light
5	Light
6	Demi Light
7	Semi Light
8	“Book” or “Text” weight
9	Semi-Bold
10	Demi-Bold
11	Bold
12	Extra Bold
13	Black
14	Extra Black
15	Ultra Black

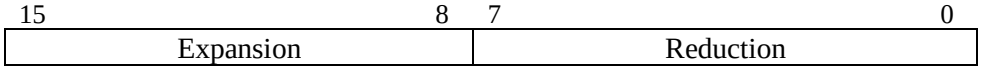
NOTE: The pseudo-bold enhancements algorithm can provide only stroke weights greater than the font's stroke weight.

DEVICE NOTE: DJ540 supports only Book/Text and Bold. Depending on the typeface and point size, DJ540 will add either one or two trailing dots (300dpi) to embolden characters.

DEVICE NOTE: LJ4V and LJ4PJ support pseudo-bolding for scalable only.

Sizing (UINT16) — Specifies the algorithmic size transformations that may be performed.

Sizing Word = Reduction + Expansion



Bit Positions	**Reduction**
0	0.5 X-dimension
1	0.5 Y-dimension
2-7	Reserved
	Expansion
8	1.5 X-dimension
9	1.5 Y-dimension
10	2 X-dimension
11	2 Y-dimension
12-15	Reserved

DEVICE NOTE: DJ540 supports only 0.5X/Y, 1.5X/Y, and 2X/Y.

DEVICE NOTE: LJ4V and LJ4PJ do not support algorithmic size transformations.

CP — Copyright Segment

This optional segment consists of ASCII data.

DP — Dual Pitch Space Character Code Segment (Format 16 only)

Dual-pitch bitmap fonts require information on the space character location for both the full-width spacing (two-byte/nominal characters) and half-width spacing (one-byte characters). This segment specifies the space character code for full-width (two-byte characters) and half width spacing (one-byte characters). This data segment is used only with Format 16 fonts.

The structure is shown below:

Byte	15 (MSB) (LSB) 0	Byte
0	DP (17488)	1
2 4	Data Segment Size	3 5
6	Full Width Character Code	7
8	Half Width Character Code	9

Full Width Character Code (UINT16) — The character code for two-byte characters.

Half Width Character Code (UINT16) — The character code for one-byte characters.

DEVICE NOTE: LJ4V and 4PJ do not support the DP segment

GC — Galley Character Segment (Format 16 only)

If an application requests a character that does not exist within the current font, the device looks in the Galley Character Segment for a substitute character to print instead. The GC segment contains the character codes of characters to be printed when specified characters are missing in the font. Different galley characters may be required for different regions of the symbol set. For example, the GC segment can be setup so an asterisk prints when a non-existent character is selected in the region 0x81-0x9F, and a question mark for characters in the region 0xE0-0xFC. The galley character segment can be downloaded with any Format 16 font, regardless of symbol set type.

If the galley character field value is 0xFFFF and the font contains a missing character glyph, the missing character glyph can be downloaded using the Download Character command (*Esc(s#W*) with a character code=0xFFFF and a glyph ID=0. For an unbound font, the symbol index specifies the character codes.

If both the character specified by the original character code and the galley character code is missing, CAP is advanced in accordance with PCL rules for missing characters — i.e. according to the current setting of CMI (Character Motion Index). If the specified character falls into more than one region, the first matching region is applied.

The GC segment is invalid if the format number is not supported or if the segment size declared in the Segment Size field is larger or smaller than required for the number of regions (N). The font download is ignored if the segment is invalid.

The GC segment definition is shown below:

Byte	15 (MSB) (LSB) 0	Byte
0	GC (18243)	
2 4	Data Segment Size (6 x n+6)	3 5
6	Format = 0	7
8	Default Galley Character	9
10	Number of Regions (N)	11
12	Region #1 Upper Left Character Code	13
14	Region #1 Lower Right Character Code	15
16	Region #1 Galley Character	17
...	...	
6*N+6	Region #N Upper Left Character Code	6*N+7
6*N+8	Region #N Lower Right Character Code	6*N+9
6*N+10	Region #N Galley Character	6*N+11

This data segment is used only with Format 16 fonts.

Default Galley Character (UINT16) — The character code of the character to be printed when a specified character is not within any of the defined regions.

Number of Regions (UINT16) — The number of regions for which galley characters are defined. Regions are defined for a table in which the first byte of the character code specifies the row, and the second byte specifies the column..

Region #x Upper Left Character Code (UINT16) — The character code defining the upper left corner of Region #.

Region #x Lower Right Character Code (UINT16) — The character code defining the lower right corner of Region #.

Region #x Galley Character (UINT16) — The character code of the character to be printed when a character within region #x is missing from the selected font.

DEVICE NOTE: GC is supported in LJ4PJ and LJ4V for TrueType only (not bitmap).

GI — Global Intellifont Data

Reserved for future use.

GT — Global TrueType Data

This data segment consists first of a Table Directory and then a series of tables used by the TrueType font scaler. Every TrueType font must have this segment. The Table Directory is patterned after the initial segment of the TrueType font file as described in *TrueType Font Files*, *Version 1.00*, Microsoft Corporation, September 1991. The Table Directory has a 12-byte header and 16 bytes per directory entry. The Table Directory is organized in alphabetical order by 4-byte table names. For each entry there is an offset relative to the beginning of the soft font's Global TrueType Data Segment. There should be exactly one entry such that the sum of its offset and its length (as given in the Table Directory) equals the specified length of the Global TrueType Data Segment.

The following tables are required for every TrueType font entity in the TrueType Data Segment, as defined in *TrueType Font Files:* **gdir, head**, **hhea**, **hmtx**, and **maxp**.

The **hmtx** table, as defined in *TrueType Font Files* contains a 4 or 2-byte datum for each "glyph" in the soft font. (There may be more glyphs than characters in a font, since composite characters may be composed of parts that do not have character status. Each part must be downloaded with its own character descriptor and character data.)

The **gdir** table should have a size and offset of 0 at the time the font definition is downloaded. The **gdir** table is built in RAM to accommodate the maximum number of glyphs to be downloaded to the given font — with 2 or 4 bytes of offset and 2 bytes of length per glyph. The maximum number of glyphs is obtained from the numGlyphs field of the **maxp** table. Entries in the **gdir** table are filled by the TrueType rasterizer as characters are downloaded.

The optional **cvt**, **fpgm** and **prep** tables, as defined in *TrueType Font Files*, will typically appear in the Global TrueType Data Segments of hinted TrueType soft fonts, but ought not to appear in unhinted fonts.

IF — Intellifont Face Data Segment

Reserved for future use.

PA — PANOSE 1 Description Segment

This variable length data segment may be used for font selection and substitution. Its definition continues to evolve. A 10-field (10-byte) version sufficient for the description of most Latin fonts appears under the OS/2 table in *TrueType Font Files*.

PF — PS-compatible Font Name Segment

This optional segment containing ASCII data is proposed as an alternative means of font selection.

TF — Typeface String Segment (Format 16 only))

This segment provides a substitute string to print for a permanent downloaded font on a PCL Typeface List. Typeface string segments can be downloaded with any Format 16 font, regardless of symbol set type.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	TF (21574)		1
2 4	Data Segment Size (2 x n+2)		3 5
6	Embedded Font Name Flag	Substitute String Length	7
8 ...	Substitute String Character List ...		9 ...

This data segment is used only with Format 16 fonts.

Embedded Font Name Flag (UBYTE) — A non-zero value in this field indicates that only the string should be printed, not the ASCII font name. A zero in this field indicates that the ASCII name of the font (from the Font Name field) should be printed in addition to the substitute string. The ASCII name may be printed in a standard font such as Line Printer, but the exact manner is product-dependent.

DEVICE NOTE: In LJ4V and LPJ4P, the ASCII name is printed in Line Printer at 16.67 pitch.

Substitute String Length (UBYTE) — The number of UINT16 characters in the Substitute String Character List.

Substitute String Character List (array of UINT16) — The characters in the substitute string. Each character is represented as an UINT16 value in the font's native mapping.

The typeface string segment is invalid if the segment size declared in the Data Segment Size field is larger or smaller than required for substitute string length or if the segment size is an odd number of bytes. The font download is ignored if the segment is invalid.

VR — Vertical Rotation Segment (Format 16 only)

The VR segment defines the lower boundary of the rotation box used when the character text path direction is set to vertical rotation. The (-1) mode of the Text Path Direction command (*Esc&c#T*) provides vertical writing compatible with Win3.1/J but does not specify whether the rotation is around CAP or some other point. Win3.1/J uses the TrueType *sTypoDescender* value to determine the point around which to rotate full-width characters when writing TrueType vertical text. The VR segment defines the lower boundary of the rotation box used in the Text Path Direction command by downloading the *sTypoDescender* value with a Format 16 font.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	VR (22098)		1
2 4	Data Segment Size		3 5
6	Format (0)		7
8	Descender value		9

Format (UINT16) — Set to 0.

Descender value (SINT16) — Defines the lower boundary of the rotation box used in the Text Path Direction command. This value should equal the "sTypoDescender" value from the "OS/2" table of the TrueType font.

If the vertical rotation segment is not downloaded with the font definition, a default value is used for Descender value. The default value is set to [Descender value = (-36/256) * ScaleFactor], where ScaleFactor is byte 64 and 65 from the Format 16 font definition.

This data segment is used only with Format 16 fonts.

DEVICE NOTE: In the LJ4PJ and LJ4V, the vertical rotation offset is computed incorrectly for certain fonts when text path direction is set to vertical (-1).

VT — Vertical Substitution Segment (Format 16 only)

The Vertical Substitution Segment contains pairs of glyph IDs. Each pair specifies the horizontal and vertical glyph ID for a character. The segment can be built directly from a TrueType **mort** table which contains a vertical substitution array. The segment definition is shown below:

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	VT (22100)		1
2 4	Data Segment Size (4*n+4)		3 5
6	Horizontal Glyph ID #1		7
8	Vertical Glyph ID #1		9
...	...		
4*N+2	Horizontal Glyph ID #N		4*N+3
4*N+4	Vertical Glyph ID #N		4*N+5
4*N+6	End of table mark #1 = 0xFFFF		4*N+7
4*N+8	End of table mark #2 = 0xFFFF		4*N+9

The **Horizontal Glyph ID** field is used by the TrueType font scaler as an ID number for the horizontal glyph data associated with a given character. The **Vertical Glyph ID** field contains the ID number for the vertical glyph data associated with the same character.

The vertical glyphs can be downloaded using the Download Character command (*Esc(s#W*) with a character code of 0xFFFF.

A **mort** table in Asian TrueType fonts typically contains a fixed-length header of 76 bytes followed by the vertical substitution array which follows the segment format described above. However, the header is designed to be of variable-length, and the location of the vertical substitution data may be elsewhere in future fonts.

This data segment is ignored if the Symbol Set Type is not Type 3 (16-bit fonts).

This data segment is used only with Format 16 fonts.

If the segment is invalid, the font download is ignored. The data segment is invalid if the value pairs are not sorted by horizontal glyph ID or if the End of table mark #1 is not 0xFFFF. The Segment Size field determines the location of the end of the table.

XW — X Windows Font Name Segment (Format 16 only)

This ASCII segment containing font names will replace the "unsanctioned" data segment written into font definitions today by Corvallis Division.

Older DeskJet Formats

DeskJets in the 5xx series and before (except DJ 540) use the following incompatible download formats. Fields that are are different than LaserJet are described afterwards.

DEVICE NOTE: DJ540 and 660 use Formats 0 and 20 (nonscalable); DJ 850 uses Formats 0, 20, and 15. The Asian versions of the 540, 660, and 850 also use format 16.

DeskJet/DeskJet Plus Bitmap Font Definition

The DeskJet/Plus bitmap definition is shown below. Grayed areas indicate fields not used by LaserJets.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (72)		1
2	Descriptor Format (DJ=5)(DJ+=9)	Symbol Set Type	3
4	Reserved (0)		5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Pitch		17
18	Height		19
20	x Height		21
22	Width Type (0)	Style	23
24	Stroke Weight	Typeface	25
26	Slant	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Spacing		33
34	Text Width		35
36	First Code		37
38	Last Code		39
40	Pitch Extended (0)	Height Extended (0)	41
42	CAP Height (0)		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Horizontal Pixel Resolution (600)		65
66	Vertical Pixel Resolution (300)		67
68	Top Double Underline Position	Top Double Underline Height	69
70	Bottom Double Underline Position	Bottom Double Underline Height	71
72	Printer Specific Block Size (20)		73
74	Font Data Size		75
76	Unidirection Flag	Compressed Flag	77
78	Reserved (0)	No Half-Pitch Flag	79
80	No Double-Pitch Flag	No Half-Height Flag	81
82	No Bold Flag	No Draft Flag	83
84	Bold Method	Reserved (0)	85
86	Pass 2 Baseline Offset		87

DeskJet 500 Bitmap Font Definition

The DeskJet 500 bitmap definition is shown below. Grayed areas indicates fields not used by LaserJets.

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (74)		1
2	Descriptor Format (12)	Symbol Set Type	3
4	Reserved		5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Pitch		17
18	Height		19
20	x Height		21
22	Width Type	Style LSB	23
24	Style MSB	Stroke Weight	25
26	Typeface		27
28	Slant	Serif Style	29
30	Quality	Placement	31
32	Underline Position	Underline Thickness	33
34	Text Height		35
36	Text Width		37
38	First Code		39
40	Last Code		41
42	Pitch Extended	Height Extended	43
44	Reserved		45
46-48	Font Number		47-49
50-64	Font Name		51-65
66	Horizontal Pixel Resolution (600)		67
68	Vertical Pixel Resolution (300)		69
70	Top Double Underline Position	Top Double Underline Height	71
72	Bottom Double Underline Position	Bottom Double Underline Height	73
74	Printer Specific Block Size (36)		75
76	Font Data Size		77
78	Unidirection Flag	Compressed Flag	79
80	Hold Time Factor	No Half-Pitch Flag	81
82	No Double-Pitch Flag	No Half-Height Flag	83
84	No Bold Flag	No Draft Flag	85
86	Bold Method	Reserved	87
88	Pass 2 Baseline Offset		89
90	Pass 3 Baseline Offset		91
92	Pass 4 Baseline Offset		93
94-108	Font Name Extension		95-109

Baseline Offsets (UINT)

For each pass, this specifies the offset from the top of the pass to the baseline of the character in dots. if this pass is not used, this value must be 0.

Bold Method (BOOL)

Specifies the algorithmic bold method to apply to this font when printing bold. A value of 0 designates the light bold method. A value of 1 designates the dark bold method. Algorithmic light bold adds 1 extra dot to the trailing edges of characters. Algorithmic dark bold adds 2 extra dots to the trailing edges of characters.

Bottom Double Underline Height (UBYTE)

Specifies the height of the bottom bar of the double underline in dots.

Bottom Double Underline Position (SBYTE)

The position of the double underline's bottom bar from the baseline.

Compressed Flag (BOOL)

A value of 1 specifies that this font has algorithmically compressed information present in its character data. A value of 0 designates that compressed width information is not present. This flag is valid only for fixed pitch fonts.

Font Data Size (UINT)

Specifies the minimum memory the font needs for a download. DeskJets ignore this field and will always attempt to store the font until space runs out; but software utilities use this number to keep track of DeskJet memory usage.

Hold Time Factor (UBYTE)

This is a font-dependent constant that provides a relative "figure of merit" for controlling the hold or dry time necessary when printing this font. Larger blacker fonts need larger Hold Time Factors.

Horizontal Pixel Resolution (UINT)

Specifies the font's horizontal pixel resolution in pixels-per-inch. This must be 600.

No Bold Flag (BOOL)

A value of 1 disables algorithmic bold for this font. A value of 0 allows algorithmic bold.

No Double Pitch Flag (BOOL)

A value of 1 disables algorithmic double pitch for this font. A value of 0 allows algorithmic double pitch.

No Draft Flag (BOOL)

A value of 1 disables draft printing for this font. A value of 0 allows draft printing.

No Half Height Flag (BOOL)

A value of 1 disables algorithmic half height for this font. A value of 0 allows algorithmic half height.

No Half Pitch Flag (BOOL)

A value of 1 disables algorithmic half pitch for this font. A value of 0 allows algorithmic half pitch.

Printer Specific Block Size (UINT)

Specifies the valid number of bytes in the printer specific block. Applicable DeskJets use a value of 20.

Slant (UBYTE)

Specifies the slant of the font.

Top Double Underline Height (UBYTE)

Specifies the height of the top bar of the double underline in dots.

Top Double Underline Position (SBYTE)

Specifies the position of double underline's top bar from the baseline.

Unidirectional Flag (BOOL)

A value of 1 specifies that this font should be printed in only one direction. A value of 0 specifies this font as a bidirectional font.

Vertical Pixel Resolution (UINT)

Specifies the font's vertical pixel resolution in pixels-per-inch. This must be 300.

10.4 Managing Fonts

Font Control *Esc * c # f/F*

Manipulates fonts and characters designated by Font ID and Character Code.

Value(#)	=	0	Delete all fonts (temporary, permanent and soft assigned)
	=	1	Delete temporary fonts (downloaded and soft assigned)
	=	2	Delete font (specified by the last Font ID)
	=	3	Delete character (last Font ID and Character Code)
	=	4	Make font temporary (specified by last Font ID)
	=	5	Make font permanent (specified by last Font ID)
	=	6	Copy/Assign the currently invoked font to RAM, make it temporary, and assign it the current Font ID
Default	=	NA	
Range	=	0 to 6 (command is ignored for other values or if no font has the specified ID)	

If the primary or secondary font is deleted, a new primary or secondary font is automatically selected from the remaining fonts.

A character becomes undefined when it is deleted (value = 3). A space is printed in place of a deleted character.

The Copy/Assign feature (value=6), called a *soft assignment*, can copy ROM or RAM fonts into RAM and assign them ID numbers. The currently selected font is copied into RAM, made temporary, and assigned the current Font ID.

- An out-of-memory during this operation deletes the newly assigned font.
- If the assigned font ID is already the same as the ID of the currently selected font, no operation takes place.
- If the assigned ID is in use and not the ID of the current font, the font with the assigned ID is deleted and the copy/assign operation takes place.
- Since scalable fonts can be rendered at any point size, a copy/assigned scalable font is not copied with a specific point size.
- Since unbound fonts can be bound to different symbol sets, a copy/assigned unbound font is not bound to a particular symbol set.

NOTE: When the Copy/Assign feature (value=6) is applied to a scalable TrueType ROM font, a font ID is assigned, but no copy is made in RAM. An attempt to delete or download characters within the font will be ignored. An attempt to delete the font will merely result in the loss of an ID number.

DEVICE NOTE: On pre-LJIII's, the page is closed and printed if the font used on the current page is deleted. HP recommends that the page not be closed. LJIII/IIID copy/assigns an unbound font by first binding it to the requested symbol set.

DEVICE NOTE: DJ5xx implements values 0,3,4,5. In DJ5xx and PJ, fonts are downloaded by default as permanent.