Chapter 16: The Color Print Model

Contents of this Chapter

This chapter discusses the following PCL commands:

16.1 Introduction

The PCL print model allows images and characters to be filled with color and patterns. Images include raster graphics, rectangular area fills (rules), font characters, and HP-GL/2 objects.

More precisely, the print model defines how source images interact with destination images through pattern, color, and transparency filters. The Logical Operations (*Esc*l#O*) command can apply logical functions such as AND, OR, XOR, NOT to any of these operands (except transparency, which must be specified first).

The print model process consists of the following steps:

1. Specify source and/or pattern transparency modes, if desired.
2. Specify the logical operation (or use the default).
3. Define the desired operands (source, destination, pattern, foreground color).

Definitions

Source: The data to be imaged. There are two kinds of sources: *mask* and *raster*. A mask source acts like a stencil whose "1" bits allow the pattern to pour through onto the destination. Source transparency mode determines whether the "0" bits are applied to the destination. Mask sources include HP-GL/2 primitives, rules, and characters.

A raster source may be specified by either the indexed or direct method. In the indexed method, each pixel identifies a palette index; in the direct method, each pixel is specified by its color components (whose color range and gamma are described by the palette). In both methods, transparency mode affects only "white" pixels.

If a pattern is involved, source pixels are logically combined with pattern pixels.

Destination: Whatever is currently defined on the page. The destination includes any images placed through previous operations.

Pattern: A rectangular area tile whose design is applied to the destination through the source. It may be a single-plane monochrome mask or a multi-plane raster color pattern. Foreground color is not applied to a downloaded color pattern. The Current Pattern (*Esc*v#T*) command can designate an active pattern, which stays in effect until changed or the printer is reset. A reset defaults the current pattern to 100% black. Rules operate differently, using patterns defined by the Fill Rectangular Area command (*Esc*c#P*).

Source Transparency Mode: Controls the transparency or opaqueness of the "white" pixels in the source image. When the mode is transparent, white pixels have no effect on the destination; when the mode is opaque, white pixels are applied to the destination.

Pattern Transparency Mode: Controls the transparency or opaqueness of the "white" pixels in the pattern. When the mode is transparent, white pixels have no effect on the destination; when the mode is opaque, white pixels are applied to the destination.

White: For the purpose of transparency modes, the meaning of "white" depends on the type of image. For characters and single-plane mask raster, a white pixel is defined to have a value of 0. For indexed raster, a white pixel is one that selects a white palette entry. For direct raster, a white pixel is one for which all color primaries meet or exceed their white reference values. Black pixels, instead of white pixels, are used for transparency in Render Algorithm 2 (*Esc*t2J*). White dots introduced in the dithering process are not subject to transparency modes; they are always opaque.

Foreground Color: Foreground color is selected by the Foreground Color command (*Esc*v#S*) from the current palette. Foreground color affects everything except color patterns and HP-GL/2 primitives. Raster color interacts with foreground color.

Texture: Texture is another name for the combination (logical AND) of pattern and foreground color, or for a downloaded color pattern (downloaded color patterns are not combined with foreground color).

Tiling: The means by which a pattern is applied to a source image. The pattern, whose upper-left pixel coincides with the *fill reference point*, is replicated horizontally and vertically across the page.

Logical Operations (ROPs): The print model allows logical operations (also called ROPs or Raster Operations by Microsoft) such as AND, OR, XOR, NOT (and combinations thereof) to be performed on source, texture, and destination. Microsoft's current version is called *ROP3*.

Rule: Rectangular area fill. Rules are created by Fill Rectangular Area (*Esc*c#P*) after specifying their vertical and horizontal size. In the PCL language, rules are a special case of source images: source transparency mode has no effect, since the rectangular area is conceptually viewed as an all 1's source. Filling a rule does not change CAP. The filled rule is not affected by end-of-line wrap, perforation skip mode, or margins. A rule may extend beyond the margins, but it will be clipped to the printable area of the logical page. Rules are not affected by raster resolution (*Esc*t#R*).

16.2 Logical Operations

The print model defines how logical operations (AND, OR, XOR, NOT, etc.) are applied to source, texture, and destination. Transparency modes and logical operations must be specified before printable data is sent.

Operators

- Source Transparency (default is transparent).
- Pattern Transparency (default is transparent).
- Logical Operators (default is Texture OR Source).

Operands

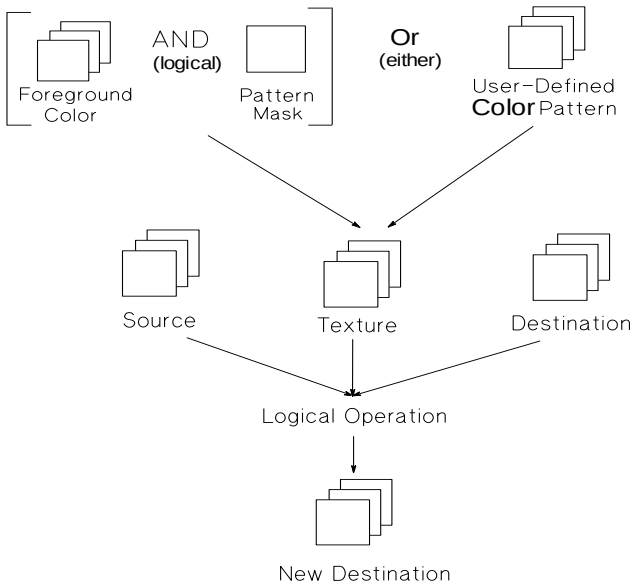
- Source objects — character cell, raster image, rule, HP-GL/2 vectors and polygons.
- Texture — foreground color + pattern mask, or color pattern (format 1).
- Destination — current page definition.

Operation

```
IF (source transparent && source == white)
    RETURN destination
IF (pattern transparent && pattern == white)
    RETURN destination
ELSE RETURN (logical operation (source, texture, destination))
```

Assuming three bits per pixel, the following diagram shows the process.

NOTE: For this discussion the colors have previously been halftoned.)



Logical Operation *Esc * l # o/O*

Specifies the logical operation to be performed in RGB color space on destination, source and texture to produce new destination data.

Value(#) = Logical operation value (see the table on the following pages)
Default = 252 (TSo)
Range = 0 to 255

This command provides 256 logical operations that map directly to Microsoft ROPs (i.e., ROP3 Raster Operations — see Volume 2, Chapter 11 of Microsoft's, *Binary and Ternary Raster Operation Codes*).

NOTE: PCL logical operations are defined for RGB color space (white = 1, black = 0). Since the printer operates in CMY and inverts the bits (white = 0, black = 1), the results may not be intuitive. ORing white with black in RGB space yields white, which is the same as ANDing in CMY space. To convert to the other color space, write the ROP in binary, invert the bits, and reverse the order.

NOTE: Logical operations are transferred when switching between PCL and HP-GL/2 contexts. HP-GL/2 uses the *MC* command to specify logical operations.

Transparency Interactions

Transparency modes and logical operations interact. The values specified by *Esc*l#O* map directly to ROP3 values only if transparency modes are explicitly set opaque (*Esc*v1N* and *Esc*v1O*). If the transparency modes are transparent (default), the following additional operations must be performed to achieve a true result.

The four basic interactions are described below. For this discussion, Source and Pattern are the transparency masks, where transparent pixels are 0's and opaque pixels are 1's.

- **Case 1:** Source and Pattern are opaque.

  Texture = Color & Pattern
  Return ROP3 (Destination, Source, Texture)

- **Case 2:** Source is opaque, Pattern is transparent.

  Texture = Color & Pattern.
  Temporary_ROP3 = ROP3 (Destination, Source, Texture)
  Image_A = Temporary_ROP3 & Not Source
  Image_B = Temporary_ROP3 & Pattern
  Image_C = (Not Pattern) & Source & Destination
  Return (Image_A | Image_B | Image_C)

- **Case 3:** Source is transparent, Pattern is opaque.

  Texture = Color & Pattern
  Temporary_ROP3 = ROP3 (Destination, Source, Texture)
  Image_A = Temporary_ROP3 & Source
  Image_B = Destination & (Not Source)
  Return (Image_A | Image_B)

- **Case 4:** Source and Pattern are transparent

  Texture = Color & Pattern
  Temporary_ROP3 = ROP3 (Destination, Source, Texture)
  Image_A = Temporary_ROP3 & Source & Pattern
  Image_B = Destination & (Not Source)
  Image_C = Destination & (Not Pattern)
  Return (Image_A | Image_B | Image_C)

The logical operations in the table on the next page are shown in RPN (reverse polish notation). Thus, the value 225 corresponds to TDSoxn, the logical function of

NOT (texture XOR (source OR destination))

The default value of this command is 252 (TSo), corresponding to a logical function of:

(texture | source)

Transparency modes and logical operations interact. The table below shows how transparency and ROP interaction change the ROP that must be specified to achieve the logical function of TSo in Case 1 (source and pattern are opaque) and Case 4 (source and patttern are transparent).

		Bits							
		7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Texture	1	1	1	1	0	0	0	0
	Source	1	1	0	0	1	1	0	0
	Destination	1	0	1	0	1	0	1	0
Case 1	**ROP3 (source & pattern are opaque)**	1	1	1	1	1	1	0	0
		(decimal 252)							
Case 4	**ROP3+Transparencies (source & pattern are transparent)**	1	1	1	0	1	0	1	0
		(decimal 234)							

Table of Logical Operations (ROPs)

The following table shows the mapping between input values and their logical operations. Note that the logical operations are specified as RPN (reverse polish notation) equations. Here is a key to describe what the Boolean Function values mean;

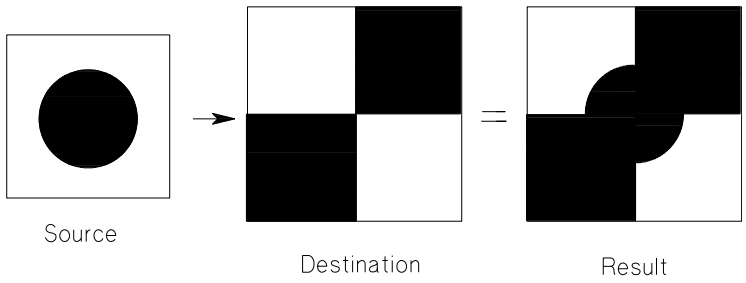
S	=	Source	a	=	AND
T	=	Texture	o	=	OR
D	=	Destination	n	=	NOT
			x	=	EXCLUSIVE OR

Input Value	Boolean Function	Input Value	Boolean Function
0	0	64	TSDnaa
1	DTSoon	65	DTSxon
2	DTSona	66	SDxTDxa
3	TSon	67	STDSanaxn
4	SDTona	68	SDna
5	DTon	69	DTSnaon
6	TDSxnon	70	DSTDaox
7	TDSaon	71	TSDTxaxn
8	SDTnaa	72	SDTxa
9	TDSxon	73	TDSTDaoxxn
10	DTna	74	DTSDoax
11	TSDnaon	75	TDSnox
12	STna	76	SDTana
13	TDSnaon	77	SSTxDSxoxn
14	TDSonon	78	TDSTxox
15	Tn	79	TDSnoan
16	TDSona	80	TDna
17	DSon	81	DSTnaon
18	SDTxnon	82	DTSDaox
19	SDTaon	83	STDSxaxn
20	DTSxnon	84	DTSonon
21	DTSaon	85	Dn
22	TSDTSanaxx	86	DTSox
23	SSTxDSxaxn	87	DTSoan
24	STxTDxa	88	TDSToax
25	SDTSanaxn	89	DTSnox
26	TDSTaox	90	DTx
27	SDTSxaxn	91	DTSDonox
28	TSDTaox	92	DTSDxox
29	DSTDxaxn	93	DTSnoan
30	TDSox	94	DTSDnaox
31	TDSoan	95	DTan
32	DTSnaa	96	TDSxa
33	SDTxon	97	DSTDSaoxxn
34	DSna	98	DSTDoax
35	STDnaon	99	SDTnox
36	STxDSxa	100	SDTSoax
37	TDSTanaxn	101	DSTnox
38	SDTSaox	102	DSx
39	SDTSxnox	103	SDTSonox
40	DTSxa	104	DSTDSonoxxn
41	TSDTSaoxxn	105	TDSxxn
42	DTSana	106	DTSax
43	SSTxTDxaxn	107	TSDTSoaxxn
44	STDSoax	108	SDTax
45	TSDnox	109	TDSTDoaxxn
46	TSDTxox	110	SDTSnoax
47	TSDnoan	111	TDSxnan
48	TSna	112	TDSana
49	SDTnaon	113	SSDxTDxaxn
50	SDTSoox	114	SDTSxox
51	Sn	115	SDTnoan
52	STDSaox	116	DSTDxox
53	STDSxnox	117	DSTnoan
54	SDTox	118	SDTSnaox
55	SDToan	119	DSan
56	TSDToax	120	TDSax
57	STDnox	121	DSTDSoaxxn
58	STDSxox	122	DTSDnoax
59	STDnoan	123	SDTxnan
60	TSx	124	STDSnoax
61	STDSonox	125	DTSxnan
62	STDSnaox	126	STxDSxo
63	TSan	127	DTSaan

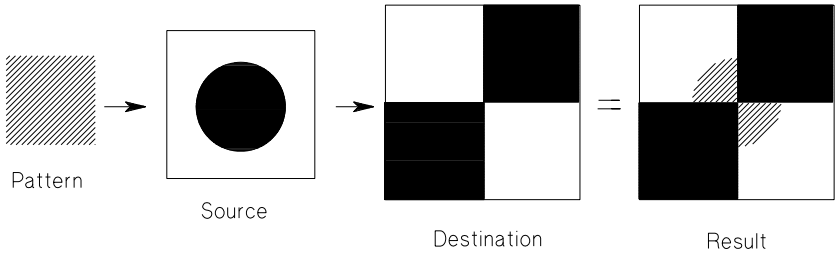
Input Value	Boolean Function	Input Value	Boolean Function
128	DTSaa	192	TSa
129	STxDSxon	193	STDSnaoxn
130	DTSxna	194	STDSonoxn
131	STDSnoaxn	195	TSxn
132	SDTxna	196	STDnoa
133	TDSTnoaxn	197	STDSxoxn
134	DSTDSoaxx	198	SDTnax
135	TDSaxn	199	TSDToaxn
136	DSa	200	SDToa
137	SDTSnaoxn	201	STDoxn
138	DSTnoa	202	DTSDxax
139	DSTDxoxn	203	STDSaoxn
140	SDTnoa	204	S
141	SDTSxoxn	205	SDTono
142	SSDxTDxax	206	SDTnao
143	TDSanan	207	STno
144	TDSxna	208	TSDnoa
145	SDTSnoaxn	209	TSDTxoxn
146	DTSDToaxx	210	TDSnax
147	STDaxn	211	STDSoaxn
148	TSDTSoaxx	212	SSTxTDxax
149	DTSaxn	213	DTSanan
150	DTSxx	214	TSDTSaoxx
151	TSDTSonoxx	215	DTSxan
152	SDTSonoxn	216	TDSTxax
153	DSxn	217	SDTSaoxn
154	DTSnax	218	DTSDanax
155	SDTSoaxn	219	STxDSxan
156	STDnax	220	STDnao
157	DSTDoaxn	221	SDno
158	DSTDSaoxx	222	SDTxo
159	TDSxan	223	SDTano
160	DTa	224	TDSoa
161	TDSTnaoxn	225	TDSoxn
162	DTSnoa	226	DSTDxax
163	DTSDxoxn	227	TSDTaoxn
164	TDSTonoxn	228	SDTSxax
165	TDxn	229	TDSTaoxn
166	DSTnax	230	SDTSanax
167	TDSToaxn	231	STxTDxan
168	DTSoa	232	SSTxDSxax
169	DTSoxn	233	DSTDSanaxxn
170	D	234	DTSao
171	DTSono	235	DTSxno
172	STDSxax	236	SDTao
173	DTSDaoxn	237	SDTxno
174	DSTnao	238	DSo
175	DTno	239	SDTnoo
176	TDSnoa	240	T
177	TDSTxoxn	241	TDSono
178	SSTxDSxox	242	TDSnao
179	SDTanan	243	TSno
180	TSDnax	244	TSDnao
181	DTSDoaxn	245	TDno
182	DTSDTaoxx	246	TDSxo
183	SDTxan	247	TDSano
184	TSDTxax	248	TDSao
185	DSTDaoxn	249	TDSxno
186	DTSnao	250	DTo
187	DSno	251	DTSnoo
188	STDSanax	252	TSo
189	SDxTDxan	253	TSDnoo
190	DTSxo	254	DTSoo
191	DTSano	255	1

16.3 The Default Print Model

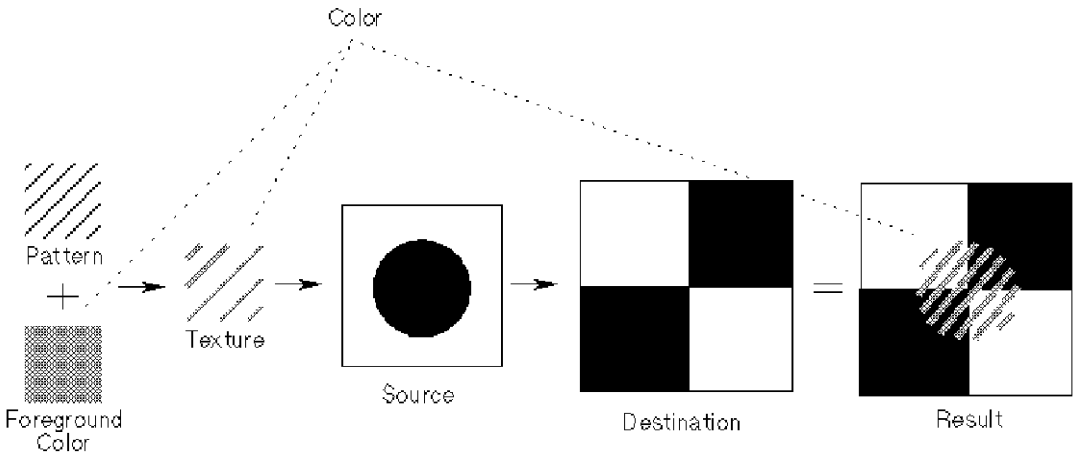
The default PCL print model is shown below. The default source and pattern transparency modes are transparent, and the default logical operation is TSo (Texture OR Source).



Source Image Applied to Destination Image
(Undefined Source pixels are transparent)



Source and Pattern applied to Destination
(Undefined Source and Pattern pixels are transparent)



Source, Pattern, and Foreground Color Applied to Destination
(Undefined Source and Pattern pixels are transparent)

16.4 Transparency Modes

Transparency modes define how white source and pattern affect the destination. White source and pattern pixels are either *transparent* and have no effect on the destination, or they are *opaque* and appear white on the destination. Pattern and foreground color do not affect white pixels.

NOTE: The meaning of "white" depends on the type of image. For characters and single-plane raster, white has a value of 0 rather than 1. For indexed raster, white is defined by the white palette entry. For direct raster, a white pixel is one whose color specification corresponds to white; for example in additive RGB spaces, all of a white pixel's primaries must meet or exceed their white reference values. White dots introduced in halftoning processes are not subject to transparency modes. Black pixels are used for transparency in Render Algorithm 2 (snap black to white and other colors to black).

Source transparency mode determines whether the source's white pixels affect the destination. Transparent white pixels have no effect; opaque white pixels appear white on the destination.

Pattern transparency mode determines whether the pattern's white pixels affect the destination. Transparent white pattern pixels have no effect; opaque white pattern pixels appear white on the destination. Non-white pattern pixels interact with non-white source pixels, and the result is applied to the destination. Foreground color is applied to the pattern's black pixels (foreground color is not applied to user-defined color patterns).

Source Transparency Mode *Esc * v # n/N*

Sets source transparency mode.

Value(#)	=	0	Transparent: white source pixels have no effect on the destination.
	=	1	Opaque: white source areas overwrite the destination.
Default	=	0	
Range	=	0,1	

A value of 0 makes the source's white areas transparent, allowing the corresponding parts of the destination image to show through. A value of 1 makes the source's white areas opaque, whiting out the corresponding parts of the destination.

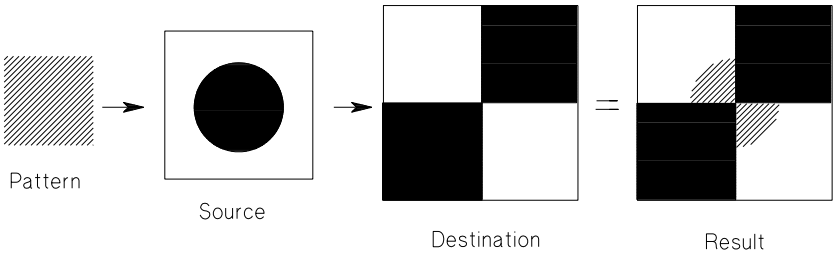
Pattern Transparency Mode *Esc * v # o/O*

Sets pattern transparency mode.

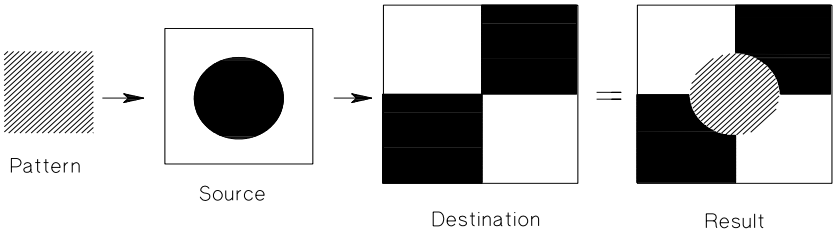
Value(#)	=	0	Transparent: white pattern areas have no effect on the destination.
	=	1	Opaque: white pattern areas overwrite the destination.
Default	=	0	
Range	=	0,1	

A value of 0 makes the pattern's white areas transparent, allowing the corresponding parts of the destination image to show through. A value of 1 makes the pattern's white areas opaque, whiting out the corresponding parts of the destination.

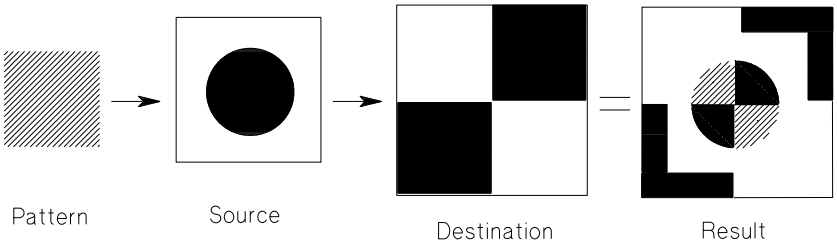
NOTE: For white rectangular area fills (rules), pattern transparency mode is always opaque.



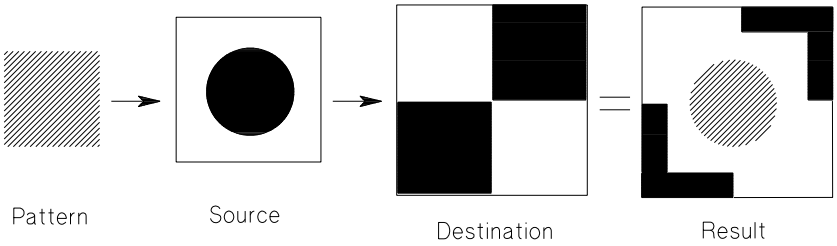
Source Transparency = 0 (Transparent)
Pattern Transparency = 0 (Transparent)



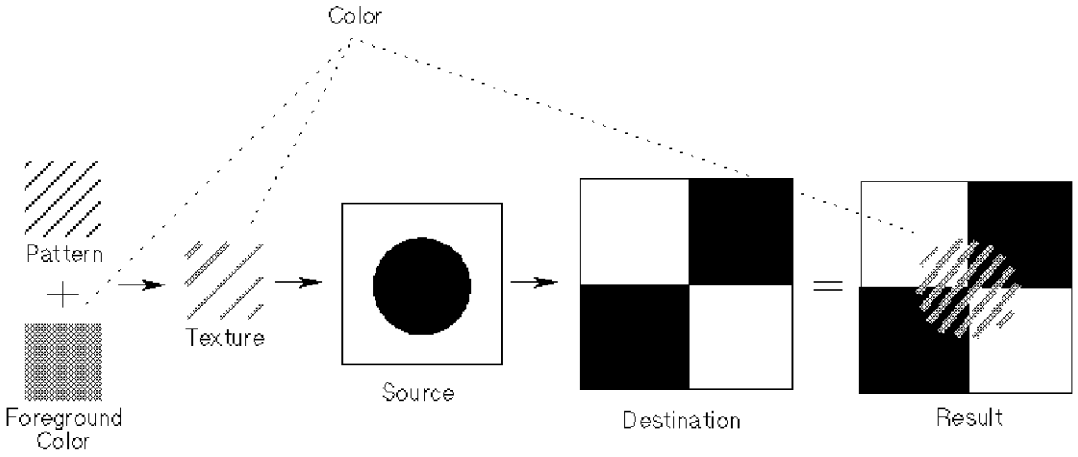
Source Transparency = 0 (Transparent)
Pattern Transparency = 1 (Opaque)



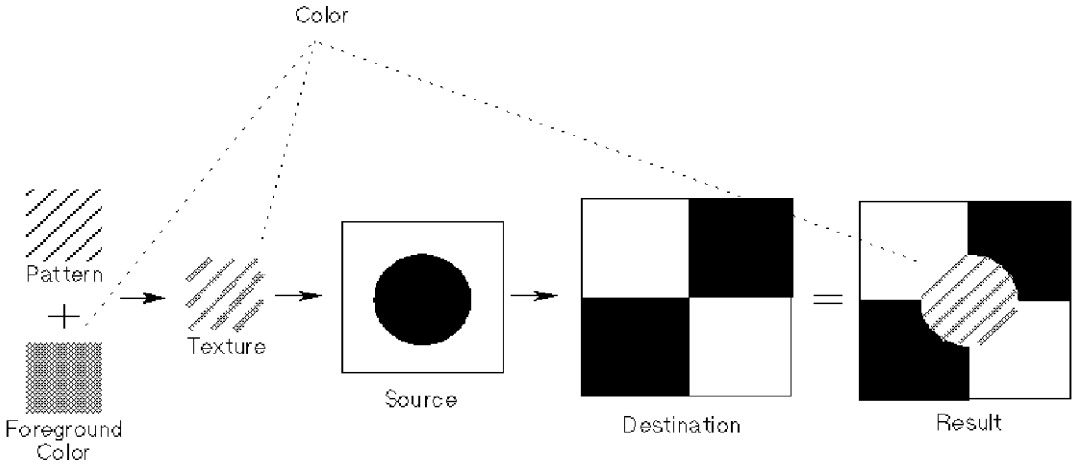
Source Transparency = 1 (Opaque)
Pattern Transparency = 0 (Transparent)



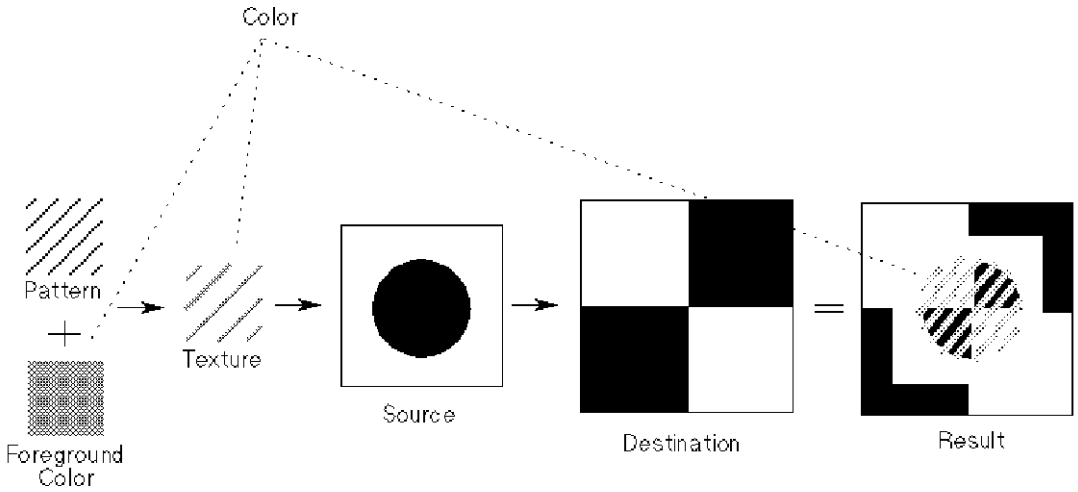
Source Transparency = 1 (Opaque)
Pattern Transparency = 1 (Opaque)



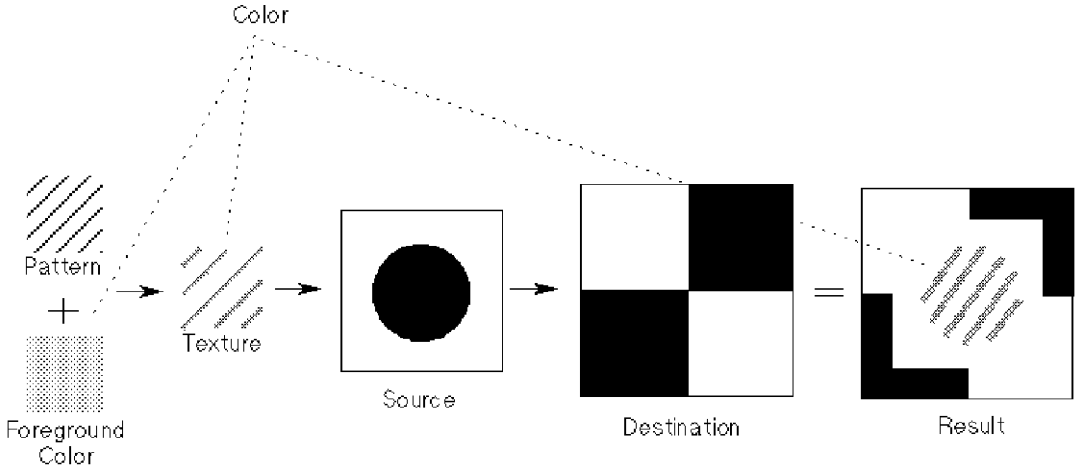
Foreground Color Applied
Source Transparency = 0, Pattern Transparency = 0



Foreground Color Applied
Source Transparency = 0, Pattern Transparency = 1



Foreground Color Applied
Source Transparency = 1, Pattern Transparency = 0



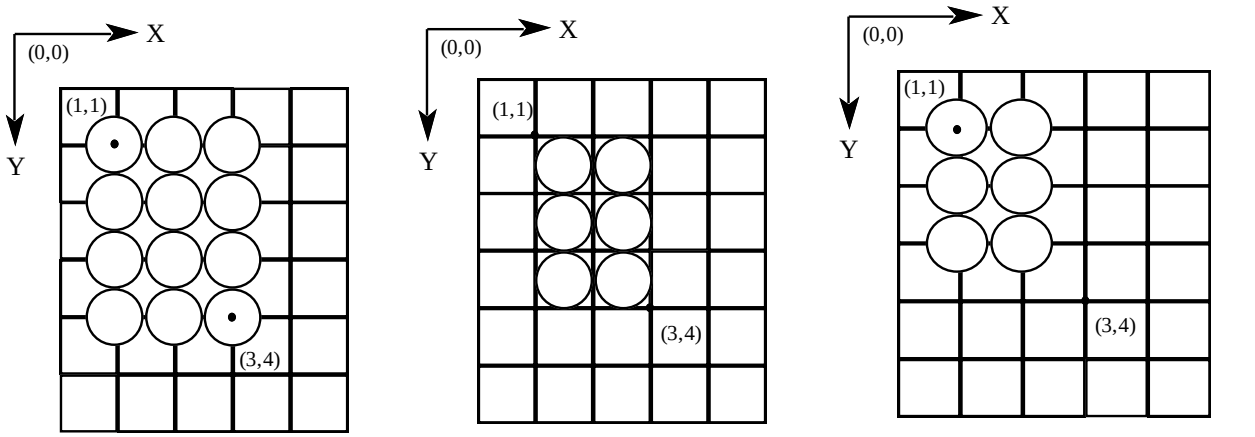
Foreground Color Applied
Source Transparency = 1, Pattern Transparency = 1

16.5 Pixel Placement

By default, the printer places pixels at the intersections of a device-dependent grid that covers the printable area on a page. When two polygons touch each other, the pixels along the common border may be printed twice or not at all — depending on the current logical operation. For example, a 1-filled source rectangle XORed with a 1-filled destination produces a 0-filled rectangle. But if another 1-filled source rectangle is placed on the page touching the first rectangle, the two destination rectangles will be 0-filled except at their common border: that is, (1^1)^1=1.

PCL provides two pixel placement models: ***grid intersection*** (the default) and ***grid centered***. Grid intersection places pixels on the grid intersections. Grid centered places pixels in the center of the grid squares, but reduces the number of rows and columns by one. The grid centered model should always be selected when two or more polygons share a common border.

In the example below, a rectangle extends from position (1,1) to (3,4). The grid centered model produces a rectangle one dot thinner and one dot shorter than the grid intersection model. Since PCL printers print only at intersections, grid centering is implemented as shown on the right.



Grid Intersection Grid Centered Actual Grid Centered Implementatio

Pixel Placement *Esc * l # r/R*

Determines how pixels are rendered in images.

Value(#)	=	0	Grid intersection
	=	1	Grid centered
Default	=	0	
Range	=	0, 1	

This command affects only HP-GL/2 polygons and PCL rules; it has no effect on characters or raster. The command can be invoked multiple times during a page; it has no effect except to switch the model used for imaging.

16.6 Patterns

As described below, the procedure for applying patterns to text and raster images is essentially the same as that used for rectangular areas (rules), except that Current Pattern (*Esc*v#T*) is used to apply the pattern to text and raster, and Fill Rectangular Area (*Esc*c#P*) is used for rules.

Patterns for Text and Raster

Use the following general procedure to fill text and raster images with a non-solid pattern.

1. Specify the Pattern ID (*Esc*c#G*).
2. Download the pattern (*Esc*c#W*). This step is for user-defined patterns only. The downloaded pattern adopts the most recently-specified pattern ID.
3. Apply the pattern to subsequent text and raster. Send the Current Pattern command (*Esc*v#T*).

Patterns for Rules (Rectangular Areas)

Use the following general procedure to full a rule with a non-solid pattern.

1. Specify the Pattern ID (*Esc*c#G*). For HP-defined patterns, select an ID that matches an HP-defined pattern.
2. Download the pattern (*Esc*c#W*). This step is for user-defined patterns only. The downloaded pattern adopts the most recently-specified pattern ID.
3. Define the rule. Position CAP and specify rule size (*Esc*c#A*, *Esc*c#H*) or (*Esc*c#B*, *Esc*c#V*).
4. Fill the rule with the pattern. Send the Fill Rectangular Area command (*Esc*c#P*).

HP-GL/2 Patterns

In HP-GL/2, patterns are downloaded by the *RF* command and applied by the *FT* or *SV* commands. HP-GL/2 may use PCL patterns; but PCL cannot use HP-GL/2 patterns.

Current Pattern *Esc * v # t/T*

Selects the current pattern fill for text and raster (not rules).

Value(#)	=	0	Solid black or foreground color
	=	1	Solid white
	=	2	HP-defined shading pattern
	=	3	HP-defined hatched pattern
	=	4	User-defined pattern
Default	=	0	
Range	=	0 to 4	

Values of 2, 3, and 4 activate the pattern specified by Pattern ID (*Esc*c#G*). The current pattern remains active even if Pattern ID is subsequently changed — that is, until a new Current Pattern command is issued.

NOTE: A pattern should be deleted (*Esc*c#Q*) after use, or this command should be sent again with a value of 0. Otherwise, the current pattern will be applied to all text and raster images.

Pattern ID *Esc * c # g/G*

Designates a unique identification number for user-defined and HP-defined patterns.

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

This command must be sent prior to downloading a user-defined pattern. When a new pattern is downloaded, any pattern already having that ID is deleted.

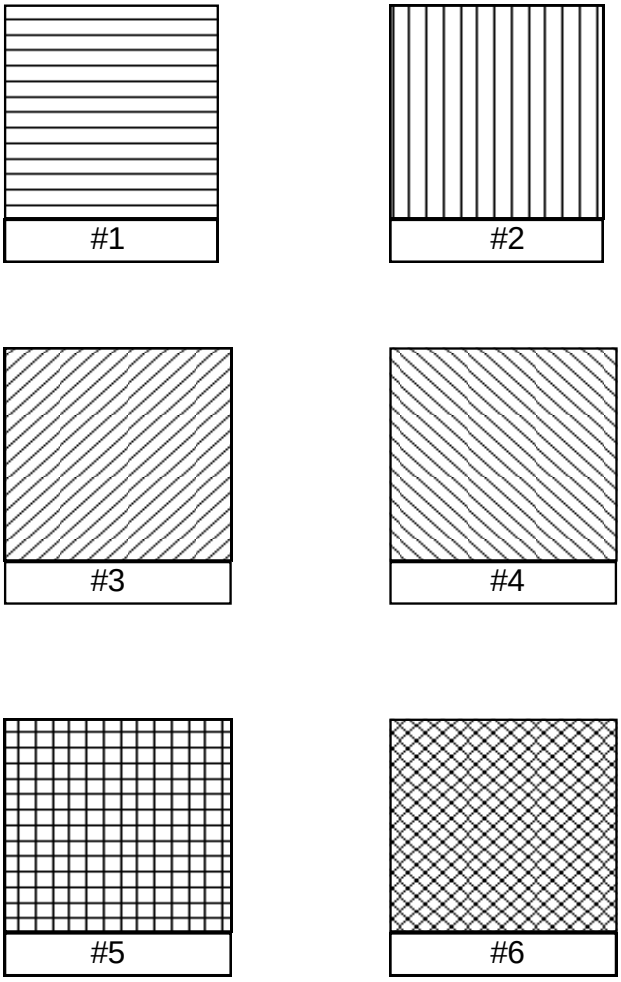
The last specified ID identifies the current pattern. If no ID exists, the Fill Rectangular Area command (*Esc*c#P*) is ignored for HP or user-defined patterns; and text and raster are represented in black or foreground color.

For HP-defined shading, IDs 1 to 100 determine the shading (nonlinear mapping of 1% to 100%). For HP-defined hatched patterns, IDs 1 to 6 select the type of hatched pattern (shown on the next page):

1	=	horizontal lines
2	=	vertical lines
3	=	diagonal lines (lower left to upper right)
4	=	diagonal lines (lower right to upper left)
5	=	cross-hatching horizontal and vertical lines
6	=	cross-hatching diagonal lines

For text and raster, Pattern ID and Current Pattern Type (*Esc*v#T*) activate an HP or user-defined pattern.

For rectangular area fills, Pattern ID and Fill Rectangular Area (*Esc*c#P*) activate an HP or user-defined pattern.



HP-defined Hatch Patterns

16.7 User-Defined Patterns

User-defined patterns, which are downloaded to the printer, are controlled by three commands:

- Download Pattern (*Esc*c#W[data]*)
- Pattern Reference Point (*Esc*p#R*)
- Pattern Control (*Esc*c#Q*)

The following three commands may contain user-defined pattern parameters and are used in conjunction with the above.

- Current Pattern (*Esc*v#T*)
- Pattern ID (*Esc*c#G*)
- Fill Rectangular Area (*Esc*c#P*)

The procedure for filling an object with a user-defined pattern is:

1. Define a binary raster image as the pattern.
2. Assign a pattern ID (*Esc*c#G*)
3. Download the pattern to the printer (*Esc*c#W[data]*)
4. To fill all subsequent text and raster with the pattern, send Current Pattern (*Esc*v#T*).
   or
5. To fill a rule, position CAP, define the size of the rule, and send position Fill Rectangular Area (*Esc*c#P*).

Download Pattern *Esc * c # W [pattern data]*

Downloads user-defined pattern data.

Value(#) = Number of bytes of pattern data
Default = 0
Range = 0 to 2^{32} -1

DEVICE NOTE: Color LJ accepts a range of 0 to 32767.

Downloaded patterns follow rules similar to downloaded fonts: they may be downloaded by ID number, deleted, and made temporary or permanent. The downloaded pattern is assigned the current Pattern ID (*Esc*c#G*). A pattern already having this ID is deleted before downloading the new pattern.

If the current pattern (specified by the last Pattern ID) is deleted, the current pattern reverts to solid black or foreground color. Subsequent text and raster objects are then represented in black or foreground color; and Fill Rectangular Area (*Esc*c#P*) is ignored for HP- or user-defined patterns.

Colors in user-defined patterns are rendered as indexes into the current palette.

For efficient memory usage, the defined pattern size should be no larger than the minimum pattern size that makes the pattern unique.

DEVICE NOTE: Pattern dimensions that are powers of 2 (e.g., 32 x 32) work most efficiently in DeskJet 1200C.

The byte-aligned binary data field is shown below. Missing data is zeroed; excess or invalid data is discarded.

Byte	15 (MSB) 8	7 0 (LSB)	Byte
0	Format	Reserved (0)	1
2	Pixel Encoding	Reserved (0)	3
4	Height in pixels		5
6	Width in pixels		7
8	X resolution*		9
10	Y resolution*		11
12	Pattern image . . .		13

*Format 20 only

Format Byte

The following two types of downloadable pattern formats are currently implemented:

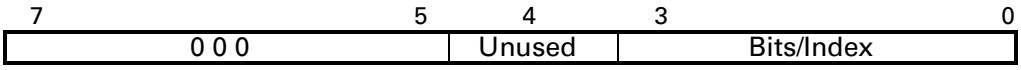
Format 0: 1 bit per pixel: black and white, or foreground color and white
Format 1: 1 or 8 bits per pixel: use current palette
Format 20: Resolution-specified. 1 bit per pixel: black and white, or foreground color and white.

Format 0 patterns have one bit per pixel. A "1" bit indicates black or foreground color. A "0" indicates either white or transparency, depending on the source and pattern transparency modes. A "0" bit cannot be colored.

Format 1 patterns use the current palette. Data is sent pixel by pixel, and the bits/index field of the pixel encoding byte determines the number of bits defining a pixel.

Format 20 adds X and Y resolution fields for devices that can specify pattern resolution.

Pixel Encoding Byte



The bits/index field may be either 1 or 8. If the value is 1, the color of each pattern dot is specified by a single bit, supporting a palette with two colors, which need not be black and white. If the value is 8, the color of each pattern dot is specified by one byte of data, allowing 256 colors. If the value of any byte is greater than the current palette size, the modulo function is applied when rendering.

NOTE: A color pattern using non-primary colors (other than black, red, green, yellow, blue, magenta, cyan, white) may interact with dithering, producing unpredictable results.

Height in Pixels

Specifies the number of raster rows in the pattern, interpreted at 300 dpi resolution. If the height is 0, the data is ignored and no pattern is defined.

DEVICE NOTE: Color LJ , DJ1200C, DJ1600C support a maximum pattern height of 32767 pixels.

Width in Pixels

Specifies the number of raster dots in the pattern, interpreted at 300 dpi resolution. If the width is 0, the data is ignored and no pattern defined.

DEVICE NOTE: Color LJ, DJ1200C, DJ1600C support a maximum pattern width of 32767 pixels.

X Resolution

Specifies horizontal resolution for printers that operate in either 300 or 600 dpi. In 600 dpi mode, a format of 0 or 1 assumes a 300 dpi pattern. For a format of 20, resolution is determined by the X and Y resolution fields. Any 300 dpi image requested while operating in 600 dpi is scaled to the correct size. The X and Y resolutions must be equal.

Y Resolution

Specifies vertical resolution for printers that operate in either 300 or 600 dpi. In 600 dpi mode, a format of 0 or 1 assumes a 300 dpi pattern. For a format of 20, resolution is determined by the X and Y resolution fields. Any 300 dpi image requested while operating in 600 dpi is scaled to the correct size. The X and Y resolutions must be equal.

Pattern Image

The pattern image is the raster data describing the pattern.

Pattern Control *Esc * c # q/Q*

Manipulates user-defined patterns.

Value(#)	=	0	Delete all patterns (temporary and permanent)
	=	1	Delete all temporary patterns
	=	2	Delete pattern (specified by last Pattern ID command)
	=	4	Designate pattern (specified by last Pattern ID command) temporary
	=	5	Designate pattern (specified by last Pattern ID command) permanent
Default	=	0	
Range	=	0 to 2,4,5	

Temporary patterns, like temporary fonts, are deleted by a reset (*EscE*).

If a pattern used on the current page is deleted, it is held internally and not disposed of until the page is printed.

If the current pattern (specified by the last Pattern ID command) is deleted, subsequent text and raster will be represented in black or foreground color, and the Fill Rectangular Area command (*Esc*c#P*) will be ignored for HP- or user-defined patterns.

Pattern Reference Point *Esc * p # r/R*

Performs two functions: (1) Sets the tiling of patterns with respect to CAP, rather than position 0,0; and (2) specifies whether the pattern rotates with print direction (*Esc&a#P*) or remains fixed.

Value(#)	=	0	Patterns are rotated with print direction
	=	1	Pattern orientation is fixed as print direction changes
Default	=	0	
Range	=	0,1	

To fill an area with a pattern, the base pattern is "tiled" or replicated across the fill area. The starting point for the tiling is called the *pattern reference point*, which is where the upper-left corner of the base pattern is positioned on the logical page.

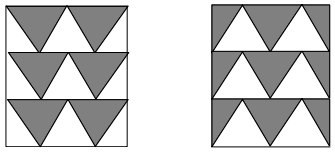
When all the tiles use the same pattern reference point, the pattern in adjoining or overlapping fragments is aligned. This command sets the reference point to CAP, allowing the pattern to be adjusted for different fill areas. The reference point may be shifted for as many fill areas as there are on a page (an area must be filled before the tile point is moved for the next fill area). This command can be used to start the pattern at a particular place in each adjoining or overlapping fragment of the fill area, regardless of alignment.

NOTE: The default pattern reference point is the upper left corner of the logical page (0,0). Unless this command is sent, the pattern is tiled with respect to position 0,0. The actual position of logical page (0,0) varies according to whether the printer feeds paper in a portrait or landscape fashion.

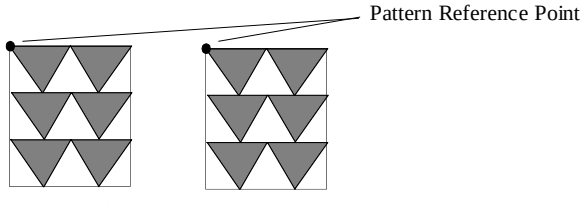
NOTE: Patterns, including user-defined patterns, are applied to images only when a fill is performed by the Fill Rectangular Area (*Esc*c#P*) or Current Pattern (*Esc*v#T*) commands, which can occur any number of times per page.

NOTE: All patterns are rotated for changes in orientation (*Esc&l#O*). When orientation is changed, the pattern is rotated but the reference point remains the same.

NOTE: The pattern reference point is not transferred to HP-GL/2, which uses the *anchor corner*.



Two areas filled (tiled) when the Pattern Reference Point is at the default (0,0) position



Two areas filled when the Pattern Reference Point is placed at the upper left corner of each area before tiling

16.8 Rectangular Area Fills (Rules)

Rules are a special case of source images: source transparency mode has no effect, since the rectangular area is conceptually viewed as an all 1's source.

Rules may be filled using patterns or textures. The current Pattern ID (*Esc*c#G*) selects the pattern and the Fill Rectangular Area command (*Esc*c#P*) tiles an area whose dimensions are specified by the Vertical and Horizontal Rectangle Size commands (*Esc*c#A*, *Esc*c#B*, *Esc*c#H*, *Esc*c#V*). A rule does not exist and cannot be printed, even though the size has been specified, until the Fill Rectangular Area command (*Esc*c#P*) is issued.

Filling a rule does not change CAP. The filled rule is not affected by end-of-line wrap, perforation skip mode, or margins. A rule may extend beyond the margins, but it will be clipped to the printable area of the logical page. Rules are not affected by raster resolution (*Esc*t#R*).

Except for the absence of undefined pixels in a rule, pattern transparency acts the same for rules as for other sources. Pattern pixels, defined and undefined, interact with the entire rectangular area.

Horizontal Rectangle Size (PCL Units) *Esc * c # a/A*

Specifies horizontal rectangle size.

Value(#) = Horizontal rectangle size in PCL Units (formerly dots)
Default = 0
Range = 0 to 2^{32} -1(clipped to logical page)

Power-up and reset default this value to 0.

Vertical Rectangle Size (PCL Units) *Esc * c # b/B*

Specifies vertical rectangle size.

Value(#) = Vertical rectangle size in PCL Units (formerly dots)
Default = 0
Range = 0 to 2^{32} -1(clipped to logical page)

Power-up and reset default this value to 0.

Horizontal Rectangle Size (Decipoints) *Esc * c # h/H*

Specifies horizontal rectangle size.

Value(#) = Horizontal rectangle size in decipoints (valid to 4 decimal places)
Default = 0
Range = 0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

Power-up and reset default this value to 0.

Vertical Rectangle Size (Decipoints) *Esc * c # v/V*

Specifies vertical rectangle size.

Value(#) = Vertical rectangle size in decipoints (valid to 4 decimal places)
Default = 0
Range = 0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

Power-up and reset default this value to 0.

Fill Rectangular Area *Esc * c # p/P*

Fills a rectangular area with the specified shade or pattern.

Value(#) = 0 Solid Black or Foreground Color
= 1 Solid White
= 2 HP-defined Shading pattern
= 3 HP-defined Hatched pattern
= 4 User-defined pattern
= 5 Current Pattern
Default = 0
Range = 0 to 5

When the value is 2, 3, or 4, the Pattern ID (*Esc*c#G*) is an index to the selected fill. A value of 5 uses the Current Pattern (*Esc*v#T*) for the fill.

When the value is 0, 2, 3, 4, or 5, the pattern transparency mode determines the effect of the pattern's undefined pixels on the destination. For a value of 1, pattern transparency is opaque.

NOTE: Pattern transparency mode is treated as if it were opaque when printing white rules.

Example: Filling Rules or Text and Raster

This example shows how to download a pattern and and fill pattern for text and raster data, then load and access another pattern for a rectangular area fill.

FILLING TEXT AND RASTER

1. Download the pattern to be used with ID #1:

*Esc*c1G*	Select a pattern ID.
*Esc*c#W[data]*	Download the pattern to be associated with the ID.

2. Activate the current pattern so it can be printed:

*Esc*v4T*	Use the pattern to fill all subsequent text and raster.

FILLING A DEFINED RULE

1. Download a pattern with ID #2:

*Esc*c2G*	Select a pattern ID.
*Esc*c#W[data]*	Download the pattern to be associated with the new ID.

2. Define a 5 x 7 dot rule.

*Esc*c5A*	Make the rule 5 dots wide.
*Esc*c7B*	Make the rule 7 dots high.

3. Fill the rule with pattern #2.

*Esc*c4P*	Use the pattern to fill the defined rule.

Note that pattern #2 is used to fill the rule; but text or raster fills will still be rendered using pattern #1.

16.9 Arbitrary Masking

Applications that benefit the most from arbitrary clipping or masking in the PCL language are those that perform gradient shading on polygons or clip images to polygons.

To fill a primitive such as a polygon with a gradient, a graphics application can theoretically use one of the following three methods:

Method	**Procedure**
Arbitrary Clip Path	1. Create a clipping window in the shape of the polygon. 2. Draw the gradient with simple rectangles (or circles in the case of a radial gradient) that fully encompass the polygon.
ROPs	1. Set ROP 90 (Destination XOR Texture) and draw the gradient with simple rectangles that fully encompass the polygon. The colors are inverted because of the XOR. 2. Set ROP 240 (Destination = Pattern) and place the solid black polygon on top of the gradient. 3. Set ROP 90 and draw the gradient again (see step 1). Gradient colors within the polygon are set to their final values and gradient colors outside the polygon are removed.
Clipping Windows	1. Create rectangular clipping windows that are the size of scanlines within the polygon to be filled. (One pixel high, and as wide as the polygon at that scanline). Notes: For radial gradients, the clipping windows may be only one pixel wide and one pixel high; This method is resolution dependent. 2. Draw a rectangle, filled with the appropriate color, that encompasses the clipping window.

The most efficient method of the three is Arbitrary Clip Path.

Clip Mask *Esc * l # p/P*

Allows the driver to create a mask that will be used on subsequent drawing primitives.

Value(#)	=	0	Turn off arbitrary masking and delete active mask from memory.
	=	1	Start mask definition.
	=	2	End definition and mask objects with the new mask. (inclusive clipping).
	=	3	End definition and mask objects with the compliment of the new mask. (exclusive clipping).
Default	=	0	
Range	=	0 to 3	

In this context, a “mask” is a bitmap created for the purpose of logical ANDing operations; and “masking” is the application of the bitmap to objects that have been rasterized into bitmap form. The mask is defined by the same objects and commands that a driver would use to draw on the page; however, none of the objects that define the mask are drawn on the page itself. Masking is performed at the printer’s bit-level resolution.

The procedure is as follows:

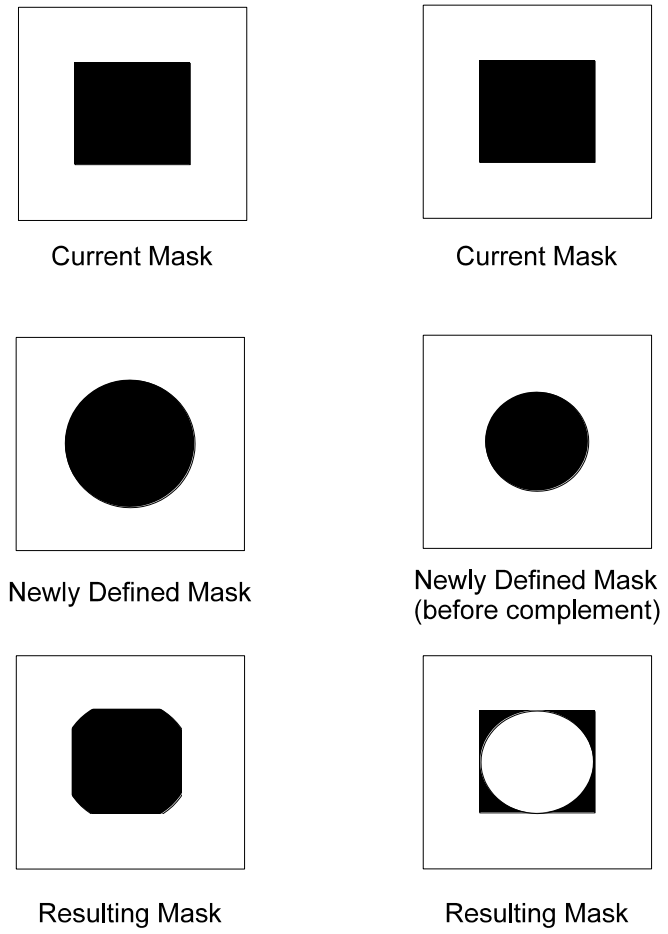
1. The driver sends this command with a value of 1 to start the mask definition.
2. The driver defines the mask by drawing objects that are rendered into the mask instead of on the page. All PCL and HP-GL/2 primitives are supported except color or multi-plane raster, patterns and color fills. The intent is to be able to use HP-GL/2 polygons, PCL text, and simple (*Esc*r1U*) single-plane monochrome (black & white) raster (bitmaps) to create a mask.
3. When the mask definition is complete, the driver initiates the masking operation by sending the command with a value of 2 or 3.
4. All subsequent objects on the page are masked until masking is turned off by a value of 0 or the printer receives a formfeed or reset. A mask may only apply to the current page.

For example, to clip a raster image to a polygon:

1. Start the definition. (Send the command with a value of 1).
2. Define the mask. (i.e., create and fill a polygon in HP-GL/2).
3. End the definition and start masking. (Send the command with a value of 2).
4. Send the image (which appears only where the polygon was drawn).
5. Terminate masking. (Send the command with a value of 0).

An existing mask is ANDed with a new mask (or its complement from operation 3). That is, primitives defining a mask will not be altered by the active mask. This feature is required to be compatible with PostScript and GDI.

If the current state of mask is off, values of 2 or 3 cause the command to be ignored. A value of 1 is ignored when already in the definition state.



Example 1 **Example 2**

In **Example 1**, a rectangular mask is already active when a new circular mask is defined. When the definition is completed with a value of 2 in the sequence, a new mask is created by ANDing together the first two masks. In **Example 2**, the same mask is active when the second mask is defined. This time, a value of 3 is sent causing the new mask to be complemented then ANDed to the first mask.

NOTE: For illustration purposes, black represents the areas where objects can mark the page.

Modifications to Current PCL Commands

When the printer is reset (via *EscE*), the arbitrary mask is disabled and deleted. While macros may use this feature, it is not part of the Modified Print Environment. State changes such as foreground color, ROP or patterns which are ignored for mask definition will be retained as appropriate for objects rendered subsequent to mask definition. Non-monochrome (single plane) raster sent during mask definition will be discarded.