5 The PCL Print Model

Introduction

The Print Model feature allows images and characters to be filled with color, with any of the printer's predefined shading or cross-hatch patterns, or with a user-defined pattern. Images include any raster graphic, such as one created with PCL raster graphics commands (as described in Chapter 6, *Raster Graphics*); a rectangular fill area (as described later in this chapter as *PCL Rectangular Area Fill Graphics*); or characters selected from any font.

Figure 5-1 illustrates the use of the print model. The following definitions are helpful in describing Print Model operation:

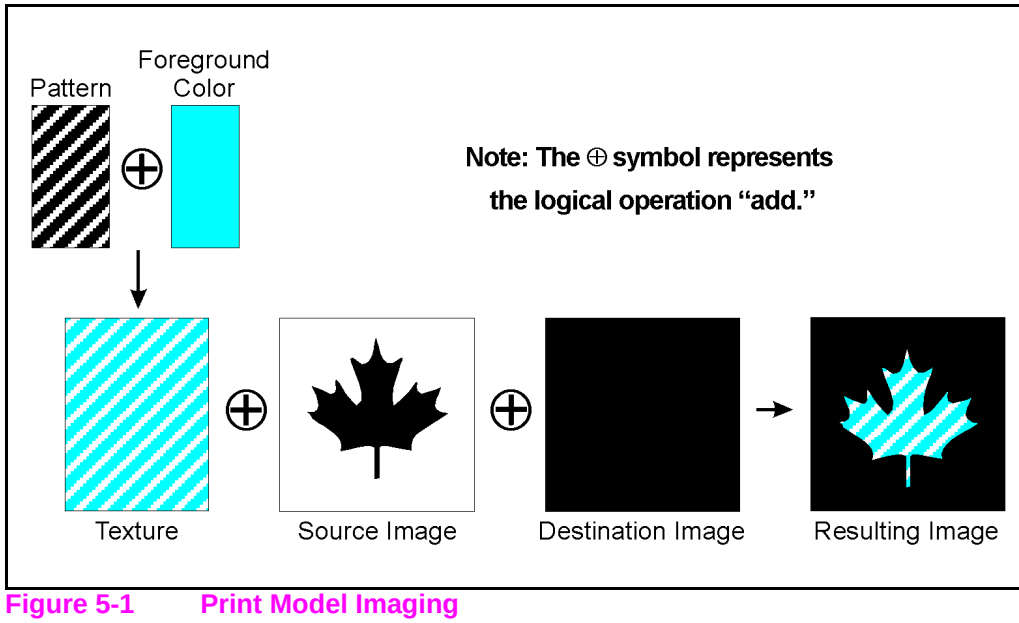

Figure 5-1 Print Model Imaging

- **Pattern**—The design which is “painted” through the non-white area of the source image onto the destination image. The pattern is defined by the Current Pattern (EC*v#T) command. It may be a color pattern or a single-plane monochrome mask, such as the printer's internal predefined shading or cross-hatch patterns, or a user-defined pattern. Foreground color is not applied to a user-defined color pattern.

  When printing a page, text and raster images are printed using the **current pattern**. Once the current pattern is specified, it stays in effect until another is selected or the printer is reset. A reset returns the current pattern to its default value (100% black). The current pattern does not always apply to rectangular area fill, which uses patterns defined by the rectangular area fill pattern commands.
- **Foreground Color**—Foreground color is selected from the current palette using the Foreground Color command (EC*v#S). Foreground color affects everything except user-defined color patterns and HP-GL/2 primitives. Raster color mixes with foreground color (see Chapter 6 “Color Raster Graphics”).
- **Texture**—Texture is another name for the combination of pattern and foreground color, or for a color pattern which is not combined with a foreground color.
- **Source Image**—the Source Image is an image in which the non-white bits are replaced by the specified pattern. The source image functions like a stencil through which the pattern is applied to the destination image. The source image may be one of the following: HP-GL/2 primitives, rules, characters, or raster images (single plane mask or multi-plane color)
- **Destination Image**—The image onto which the source image/texture combination is placed. The destination image includes any images placed through previous operations.
- **Source Transparency Mode**—The transparency or opaqueness of the source image’s “white” pixels as they are applied to the destination image (see the note below). Setting the source transparency mode to 1 (opaque) applies the source image's white pixels to the destination image; with a setting of 0 (transparent), these pixels have no effect on the destination.

- **Pattern Transparency Mode**—The transparency or opaqueness of the “white pixels” in the pattern (see the note below). When set to 0 (transparent), these pixels have no effect on the destination; when set to 1 (opaque), they are applied through the black pixels of the source pattern to the destination.
- **Logical Operations**—the Print Model uses logical operations, such as AND, OR, XOR, and NOT when determining which bits of the source, pattern, and texture become part of the resulting image. The Logical Operations command (EC*l#O) can vary the logical operation used, thus varying the outcome.

Note

For RGB color images, “white” pixels are those for which all color primaries are 255. For CMY color images, “white” pixels are those for which all color primaries are 0.

For all rendering algorithms, white dots introduced in the dithering process are not subject to transparency modes.

Figure 5-2 illustrates the effects of the source and pattern transparency modes on the final image. (The transparency modes work a little differently with rectangular area fill—see “Pattern Transparency for Rectangular Area Fill” near the end of this chapter.)

This example uses the default ROP. The output may appear differently depending on the colors used.

Figure 5-2 Opaque and Transparency Modes

Figure 5-3 demonstrates the transparency modes. In the first example (1a), the transparency mode for both the source image and the pattern is transparent. Since the source mode is “transparent,” only the non-white region (the circle) of the source image is overlaid on the destination. Since the pattern mode is also transparent, the patterned source image is applied only to the white areas of the destination.

In the second example (1b), the source mode is still “transparent,” but the pattern mode is “opaque” – so the pattern's white pixels are applied to the destination. The resulting image shows the entire circle region visible and patterned.

In the third example (1c), the source mode is “opaque” and the pattern mode is transparent. Since the source mode is opaque, the entire source image (the circle and the surrounding square) appears overlaid onto the destination. The pattern, however, is allowed to pour through only onto the white-pixeled area of the destination. The circle is visible in the result, but only two opposing quarters appear patterned.

In the fourth example (1d), both source and pattern modes are “opaque.” The entire source image is overlaid onto the destination, and the entire circle is patterned.

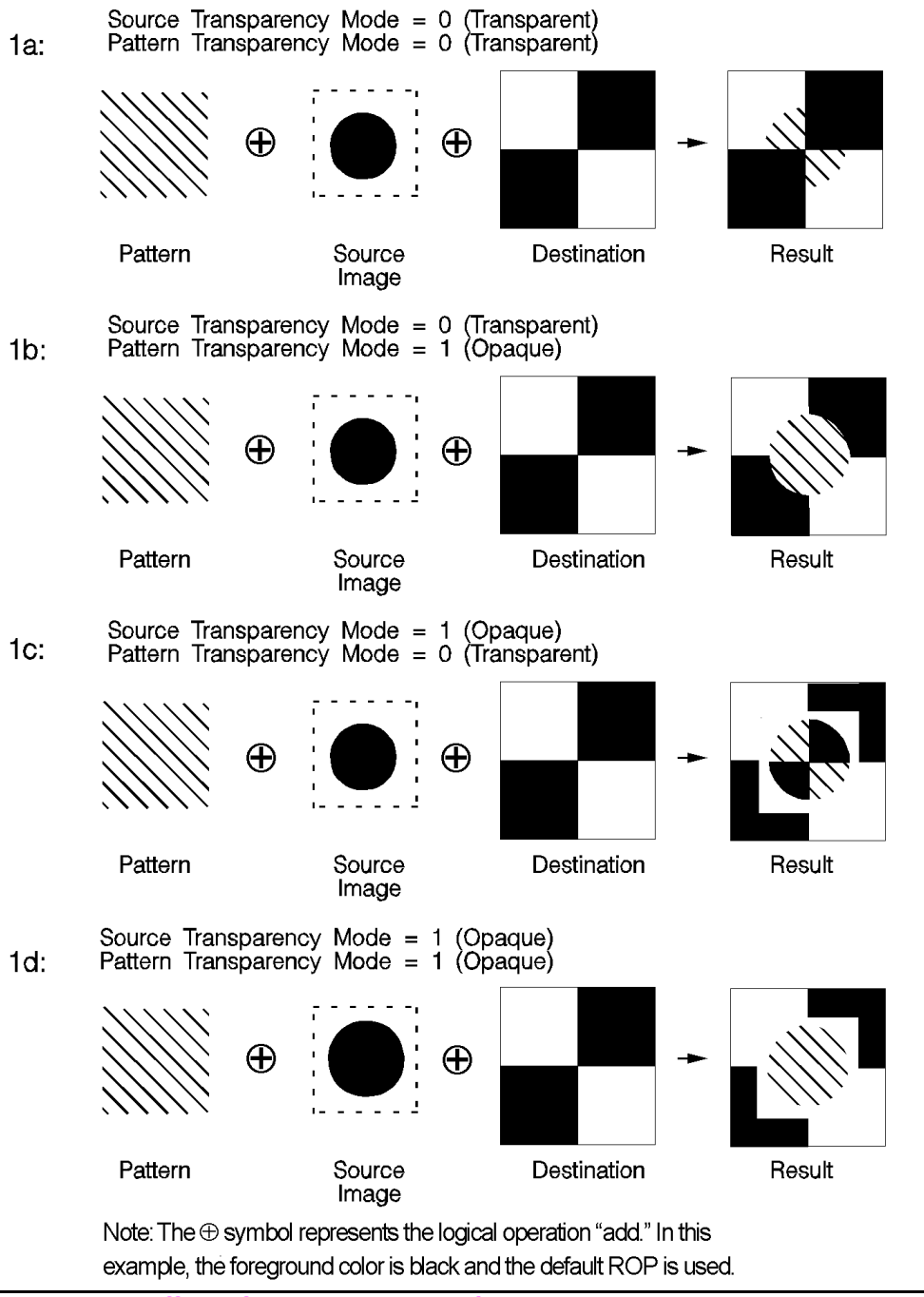

Figure 5-3 Effect of Transparency Modes on Images

Command Sequence

The table below shows the Print Model command sequence for selecting a current pattern and using it to fill a destination image. The commands for specifying transparency modes, logical operations, and patterns are discussed beginning on the following page. Foreground color is described in Chapter 3.

Operation	Comments
Download Page Data	Prior raster and character data downloaded to the page is considered destination image.
Select Transparency Modes	^{E}C*v#N (source) and/or ^{E}C*v#O (pattern)
Specify the Logical Operation	If a logical operation other than the default (TSo-252) is desired, specify the operation with the ^{E}C*ℓ#O command.
Select Specific Pattern ID	Pattern ID ^{E}C*c#G
Download User-Defined Pattern	If using a user-defined pattern, it must be downloaded to the printer before using it.
Select Pattern	^{E}C*v4T (selects downloaded pattern)
Specify the Foreground Color	For color printers, specify a Foreground Color (^{E}C*v#S) if desired. (This step is unnecessary if a color pattern is used.)
Download Source Image Data	Raster image/characters
Return to regular print mode	Default *current* pattern and transparency modes: ^{E}C*v0T (100% black pattern selected) and ^{E}C*v0N ^{E}C*v0O (transparency modes selected).
Download remaining page data	Transfer data for regular printing, or the above process may be repeated to produce another print model effect.
End of Page Data	

Source Transparency Mode Command

The Select Source Transparency Mode command sets the source image's transparency mode to transparent or opaque. This command determines whether the source's white pixels are applied to the destination.

$^{E}_{C}$ * **v # N**

# =	0	-	Transparent
	1	-	Opaque

Default	=	0
Range	=	0, 1 (other values cause the command to be ignored)

With a transparency mode of “0” (transparent), the white regions of the source image are not copied onto the destination. With a transparency mode of “1” (opaque), the white pixels in the source are applied directly onto the destination. White pixels are unaffected by pattern or foreground color; they are either white or transparent.

Note

For RGB color images, “white” pixels are those for which all color primaries are 255. For CMY color images, “white” pixels are those for which all color primaries are 0.

White dots introduced in the dithering process are not subject to transparency modes.

Refer to the preceding definitions and the discussion of Figure 5-3 for an explanation of the effects of source transparency.

Pattern Transparency Mode Command

The Pattern Transparency Mode command sets the pattern's transparency mode to transparent or opaque.

$^{E}_{C}$ * **v # O**

# =	0	-	Transparent
	1	-	Opaque

Default	=	0
Range	=	0, 1 (other values cause the command to be ignored)

A transparency mode of “0” (transparent) means that the white regions of the pattern image are not copied onto the destination. A transparency mode of “1” (opaque) means that the white pixels in the pattern are applied directly onto the destination.

Note

When printing white rules, the pattern transparency is treated as if it were “opaque”; white rules erase black rules regardless of the transparency mode.

For RGB color images, “white” pixels are those for which all color primaries are 255. For CMY color images, “white” pixels are those for which all color primaries are 0.

White dots introduced in the dithering process are not subject to transparency modes.

Refer to the preceding definitions and the discussion of Figure 5-2 and Figure 5-3 for an explanation of the effects of pattern transparency.

Logical Operations

The basic print model defines how a pattern, source image, and destination image are applied to each other using the print model's transparent and opaque modes to produce a resulting image. The Logical Operations (E_C*ℓ#O) command specifies which logical operation is to be performed on the source, texture, and destination to produce a new destination. Transparency modes should be specified before the logical operation is performed or printable data is sent.

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).

Definitions

Source: The source image may be one of the following:

- HP-GL/2 primitives
- Rules
- Characters
- Raster images (single plane mask or multiplane color)

Destination: The destination image contains whatever is currently defined on the page. It includes any images placed through previous operations.

Pattern or Texture: The pattern is defined by the Select Current Pattern command (E_C*v#T). The terms pattern and texture are used interchangeably in this section.

Transparency Modes: The white pixels of the source and/or pattern may be made transparent (source transparency 0, pattern transparency 0). The destination shows through these areas. Transparency modes are set by the Source Transparency (E_C*v#N) and Pattern Transparency (E_C*v#O) commands.

The Print Model allows logical operations, such as AND, OR, XOR, NOT, to be performed on source, texture, and destination images. Transparency modes and Logical Operation must be specified before printable data is sent.

Operators

- Source Transparency (specified before logical operation; default is transparent)
- Pattern Transparency (specified before logical operation; 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, color pattern (format 1).
- Destination: current page definition

Operation

- IF (source transparent && source == white) RETURN destination
- IF (pattern transparent && pattern == white && source != white) RETURN destination
- ELSE RETURN (logical op (source, texture, destination)

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

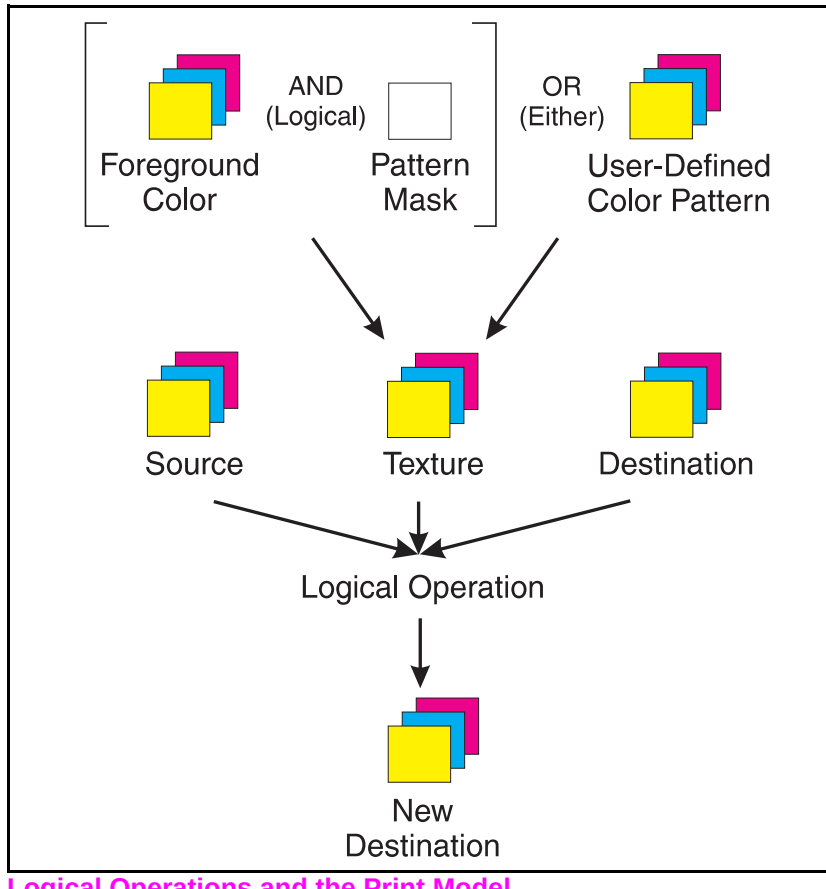

Figure 5-4 **Logical Operations and the Print Model**

Note

The Logical Operation command (Ec*l#O) provides 255 possible logical operations. All of these logic operations map directly to their ROP3 (raster operation) counterparts (see the Microsoft Document, Reference, Volume 2, Chapter 11, Binary and Ternary Raster Operation Codes).

The logical operations were defined for Microsoft Windows for an RGB color space. In RGB space, a “1”' is white and a “0” is black.

Logical Operations and Transparency Interactions

As described above, transparency modes operate in addition to logical operations. The Logical Operations (ROP3) in Table 5-4 are true only if source and pattern transparency (for white pixels) are explicitly set to opaque (^{E}C*v1N and ^{E}C*v1O). If source and/or pattern transparency modes are transparent (defaulted), the additional operations shown below must be performed to achieve the final result.

The four basic interactions are:

- **Case 1:** Source and Pattern are opaque.
  Texture = Color & Pattern.
  RETURN ROP3 (Dest, Src, Texture).
- **Case 2:** Source is opaque, Pattern is transparent.
  Texture = Color & Pattern.
  Temporary_ROP3 = ROP3 (Dest, Src, Texture).
  Image_A = Temporary_ROP3, & Not Src.
  Image_B = Temporary_ROP3 & Pattern.
  Image_C = Not Pattern & Src & Dest.
  RETURN Image_A | Image_B | Image_C
- **Case 3:** Source is transparent, Pattern is opaque.
  Texture = Color & Pattern.
  Temporary_ROP3 = ROP3 (Dest, Src, Texture).
  Image_A = Temporary_ROP3 & Src.
  Image_B = Dest & Not Src.
  RETURN Image_A | Image_B
- **Case 4:** Source and Pattern are transparent
  Texture = Color & Pattern.
  Temporary_ROP3 = ROP3 (Dest, Src, Texture).
  Image_A = Temporary_ROP3 & Src & Pattern.
  Image_B = Dest & Not Src.
  Image_C = Dest & Not Pattern.
  RETURN Image_A | Image_B | Image_C.

Note The Transparency Mode is applied based on the color of each pixel. However, the Logical Operation is applied on a bit-by-bit basis without regard to color. In order to obtain a result consistent with the Logical Operation, the transparency modes should be set to Source Opaque and Pattern Opaque. In order to obtain a result consistent with the desired transparency mode, the Logical Operation should be set to 252 and the foreground color set to black.

Logical Operation Command

Specifies the logical operation (ROP) to be performed in RGB color space on destination, source and texture to produce new destination data. Texture is defined as a combination of pattern and foreground color.

$^{E}{}_{C}$***l # O**

= Logical operation value (see Table 5-4)

Default = 252 (TSo)
Range = 0 to 255

The Logical Operation code, or Raster OPeration (ROP) code, is simply a systematic method of encoding all of the 256 possible ways that a Texture, Source, and Destination can be combined. Table 5-4 gives a table of ROPs from ROP 0 to ROP 255, where each operation is defined as a logic equation. This table can be difficult to understand and use. Furthermore, it does not show the differences that depend on the color space. A truth table is an alternative method for understanding the results of a logical operation. When used with ROPs for finding the resulting destination value, it is more easily understood than the logic operation.

ROP_{RGB} 252 (11111100)[1]			
(T)exture	**(S)ource**	**(D)estination**	**(D)estination**
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

[1] The first destination column is ignored and the second destination column is the result of the ROP (White = [1,1,1] and Black = [0,0,0]).

For example, the logic equation for ROP 252 in the RGB color space is T OR S, which is shown as TSo in Table 5-4. The truth table for the ROP is shown above, and can be seen to correspond to the logic equation TSo, that is, D gets the value of T OR S without regard to the current value of D. Furthermore, the binary value of 252 is 11111100 and corresponds with the value of the D for all the combinations of T and S, when the truth table starts with (1, 1, 1) and ends with (0, 0, 0).

It's possible to derive the logical operation for a truth table and to create a truth table for a logical operation. However, the most important point is that the binary value of the ROPs number gives the Destination for all possible combinations of Texture, Source, and Destination.

The way the bits of the ROPs number map to the combinations of Texture, Source, and Destination depends on whether the color space is RGB or CMY. The least significant bit of the RGB ROP value maps to (0, 0, 0), the color black in RGB, and the most significant bit to (1, 1, 1), white in RGB. On the other hand, the CMY ROP reverses the mapping. This reversal hinges on the fact that RGB and CMY are the inverse of each other, i.e., RGB Black is (0, 0, 0) and CMY Black is (1, 1, 1), white. All other colors show the same relationship.

ROPs in the RGB Color Space

The RGB ROP truth tables shown in Table 5-1 illustrate how ROP 252 and ROP 90 work, and most importantly how the bits in the ROP map show destination values for each combination of Texture, Source and Destination. A “1” in the RGB color space represents white and a “0” black, which makes determining what shows on paper cumbersome for users since the paper is marked when the Destination has a “0” value.

Table 5-1. RGB ROP Truth Tables

ROP$_{rgb}$ 252					**ROP$_{rgb}$ 90**					**ROP$_{rgb}$ *n***				
(11111100)					**(01011010)**					**n=($b_7b_6b_5b_4b_3b_2b_1b_0$)**				
T	**S**	**D**	**D**		**T**	**S**	**D**	**D**		**T**	**S**	**D**	**ROP *n***	**D**
1	1	1	1	white	1	1	1	0	white	1	1	1	b_7	b_7
1	1	0	1		1	1	0	1		1	1	0	b_6	b_6
1	0	1	1		1	0	1	0		1	0	1	b_5	b_5
1	0	0	1		1	0	0	1		1	0	0	b_4	b_4
0	1	1	1		0	1	1	1		0	1	1	b_3	b_3
0	1	0	1		0	1	0	0		0	1	0	b_2	b_2
0	0	1	0		0	0	1	1		0	0	1	b_1	b_1
0	0	0	0	black	0	0	0	0	black	0	0	0	b_0	b_0

ROPs in the CMY Color Space

The CMY ROP truth tables in Table 5-2 shows examples of how the ROPs number determines the value of the Destination for all combinations of Texture, Source, and Destination. In the CMY color space a “0” is the absence of ink (white) and a “1” is the presence of ink (black), the opposite of the RGB color space value for black and white. Therefore, the ROPs results (Destination values) for the CMY color space are the opposite (negation) of the RGB values. However, a CMY ROP is easier to use when determining if the page is marked, since a “1” denotes marking.

Table 5-2. CMY ROP Truth Tables

ROP_{cmy} 252 (00000011)					ROP_{cmy} 90 (10100101)					ROP_{cmy} *n* $n=(b_7b_6b_5b_4b_3b_2b_1b_0)$				
T	**S**	**D**	**D**		**T**	**S**	**D**	**D**		**T**	**S**	**D**	**ROP *n***	**D**
0	0	0	0	white	0	0	0	1	white	0	0	0	b_7	!b7
0	0	1	0		0	0	1	0		0	0	1	b_6	!b6
0	1	0	0		0	1	0	1		0	1	0	b_5	!b5
0	1	1	0		0	1	1	0		0	1	1	b_4	!b4
1	0	0	0		1	0	0	0		1	0	0	b_3	!b3
1	0	1	0		1	0	1	1		1	0	1	b_2	!b2
1	1	0	1		1	1	0	0		1	1	0	b_1	!b1
1	1	1	1	black	1	1	1	1	black	1	1	1	b_0	!b0

Using a ROP

The first step in using a ROP is to determine which color space you're in: RGB or CMY. Then determine the binary value of the ROP used. For example, suppose you want to use ROP 90 in the CMY color space. The binary equivalent of 90 is 01011010 when written in most significant to least significant bit order.

Looking at the truth table for ROP 90 in Table 5-2 you can see that the only time the page is marked is when the Texture and Destination are both “0” or both “1.” However, the same result is given by negating each bit of the ROP number, 90, to give 10100101. Using the general table for CMY ROPs (the rightmost table in Table 5-2) you can plug the bit values from 90 into b_7 through b_0 to obtain the values in the truth table for ROP 90. Similarly, using ROP 90 in the RGB color space entails plugging 01011010 in the general table for RGB ROPS (the rightmost table in Table 5-1) to obtain the values in the truth table for ROP_{rgb} 90 (also in Table 5-1). This process works for any value from 0 to 255 and can be used to determine what will show for any given ROP, in either the RGB or CMY color spaces.

Note

Since PCL logical operations are interpreted in RGB space (white = 1, black = 0) rather than in CMY space (white = 0, black = 1), the results may not be intuitive. For example, ORing a white object with a black object in RGB space yields a white object. This is the same as ANDing the two objects in CMY space. It must be remembered that the printer operates in CMY space and inverts the bits. To convert from one color space to the other, write the ROP in binary format, invert the bits, and reverse the order.

When source and/or pattern transparency modes are set opaque (not defaulted), values specified by this command map directly to the ROP3 (raster operation) table values on the following page. However, when source and/or pattern transparency modes are set transparent, the additional operations shown on the previous page must be performed to achieve the final result.

Logical operations in the table are shown in RPN (reverse polish notation). For example, the value 225 corresponds to TDSoxn, the logical function of:

NOT (texture XOR (destination OR source))

Note

^{E}c***ℓ*** **#** **O** is the PCL Version of the HP-GL/2 MC command.

This command sets the ROP value which affects not only PCL operation but also the HP-GL/2 ROP value.

EXAMPLE

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

(texture | source)

The result is computed below (source and pattern opaque).

Table 5-3. Logical Operation (ROP3)

	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
ROP3 (source & pattern)	1	1	1	1	1	1	0	0
	(decimal 252)							

Each column of destination, source, and texture values are the input to the logical function. The result, 252, is the value that would be sent to identify the logical operation (refer to page 5-12 for source/pattern transparency interactions).

Table of Logical Operations

The Logical Operations (ROP3) table (Table 5-4) 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

Note Since logical operations are interpreted in RGB space (white = 1 and black = 0) rather than in CMY space (white = 0 and black = 1), the results may not be intuitive. For example, ORing a white object with a black object in RGB space yields a white object. This is the same as ANDing the two objects in CMY space. It must be remembered that the printer operates in something similar to a CMY space and inverts the bits and reverses the order.

Table 5-4. Logical Operations (ROP3)

Input Value	Boolean Function	Input Value	Boolean Function
0	0	27	SDTSxaxn
1	DTSoon	28	TSDTaox
2	DTSona	29	DSTDxaxn
3	TSon	30	TDSox
4	SDTona	31	TDSoan
5	DTon	32	DTSnaa
6	TDSxnon	33	SDTxon
7	TDSaon	34	DSna
8	SDTnaa	35	STDnaon
9	TDSxon	36	STxDSxa
10	DTna	37	TDSTanaxn
11	TSDnaon	38	SDTSaox
12	STna	39	SDTSxnox
13	TDSnaon	40	DTSxa
14	TDSonon	41	TSDTSaoxxn
15	Tn	42	DTSana
16	TDSona	43	SSTxTDxaxn
17	DSon	44	STDSoax
18	SDTxnon	45	TSDnox
19	SDTaon	46	TSDTxox
20	DTSxnon	47	TSDnoan
21	DTSaon	48	TSna
22	TSDTSanaxx	49	SDTnaon
23	SSTxDSxaxn	50	SDTSoox
24	STxTDxa	51	Sn
25	SDTSanaxn	52	STDSaox
26	TDSTaox	53	STDSxnox

Table 5-4. Logical Operations (ROP3) (continued)

Input Value	Boolean Function	Input Value	Boolean Function
54	SDTox	81	DSTnaon
55	SDToan	82	DTSDaox
56	TSDToax	83	STDSxaxn
57	STDnox	84	DTSonon
58	STDSxox	85	Dn
59	STDnoan	86	DTSox
60	TSx	87	DTSoan
61	STDSonox	88	TDSToax
62	STDSnaox	89	DTSnox
63	TSan	90	DTx
64	TSDnaa	91	DTSDonox
65	DTSxon	92	DTSDxox
66	SDxTDxa	93	DTSnoan
67	STDSanaxn	94	DTSDnaox
68	SDna	95	DTan
69	DTSnaon	96	TDSxa
70	DSTDaox	97	DSTDSaoxxn
71	TSDTxaxn	98	DSTDoax
72	SDTxa	99	SDTnox
73	TDSTDaoxxn	100	SDTSoax
74	DTSDoax	101	DSTnox
75	TDSnox	102	DSx
76	SDTana	103	SDTSonox
77	SSTxDSxoxn	104	DSTDSonoxxn
78	TDSTxox	105	TDSxxn
79	TDSnoan	106	DTSax
80	TDna	107	TSDTSoaxxn

Table 5-4. Logical Operations (ROP3) (continued)

Input Value	Boolean Function	Input Value	Boolean Function
108	SDTax	135	TDSaxn
109	TDSTDoaxxn	136	DSa
110	SDTSnoax	137	SDTSnaoxn
111	TDSxnan	138	DSTnoa
112	TDSana	139	DSTDxoxn
113	SSDxTDxaxn	140	SDTnoa
114	SDTSxox	141	SDTSxoxn
115	SDTnoan	142	SSDxTDxax
116	DSTDxox	143	TDSanan
117	DSTnoan	144	TDSxna
118	SDTSnaox	145	SDTSnoaxn
119	DSan	146	DTSDToaxx
120	TDSax	147	STDaxn
121	DSTDSoaxxn	148	TSDTSoaxx
122	DTSDnoax	149	DTSaxn
123	SDTxnan	150	DTSxx
124	STDSnoax	151	TSDTSonoxx
125	DTSxnan	152	SDTSonoxn
126	STxDSxo	153	DSxn
127	DTSaan	154	DTSnax
128	DTSaa	155	SDTSoaxn
129	STxDSxon	156	STDnax
130	DTSxna	157	DSTDoaxn
131	STDSnoaxn	158	DSTDSaoxx
132	SDTxna	159	TDSxan
133	TDSTnoaxn	160	DTa
134	DSTDSoaxx	161	TDSTnaoxn

Table 5-4. Logical Operations (ROP3) (continued)

Input Value	Boolean Function	Input Value	Boolean Function
162	DTSnoa	189	SDxTDxan
163	DTSDxoxn	190	DTSxo
164	TDSTonoxn	191	DTSano
165	TDxn	192	TSa
166	DSTnax	193	STDSnaoxn
167	TDSToaxn	194	STDSonoxn
168	DTSoa	195	TSxn
169	DTSoxn	196	STDnoa
170	D	197	STDSxoxn
171	DTSono	198	SDTnax
172	STDSxax	199	TSDToaxn
173	DTSDaoxn	200	SDToa
174	DSTnao	201	STDoxn
175	DTno	202	DTSDxax
176	TDSnoa	203	STDSaoxn
177	TDSTxoxn	204	S
178	SSTxDSxox	205	SDTono
179	SDTanan	206	SDTnao
180	TSDnax	207	STno
181	DTSDoaxn	208	TSDnoa
182	DTSDTaoxx	209	TSDTxoxn
183	SDTxan	210	TDSnax
184	TSDTxax	211	STDSoaxn
185	DSTDaoxn	212	SSTxTDxax
186	DTSnao	213	DTSanan
187	DSno	214	TSDTSaoxx
188	STDSanax	215	DTSxan

Table 5-4. Logical Operations (ROP3) (continued)

Input Value	Boolean Function	Input Value	Boolean Function
216	TDSTxax	236	SDTao
217	SDTSaoxn	237	SDTxno
218	DTSDanax	238	DSo
219	STxDSxan	239	SDTnoo
220	STDnao	240	T
221	SDno	241	TDSono
222	SDTxo	242	TDSnao
223	SDTano	243	TSno
224	TDSoa	244	TSDnao
225	TDSoxn	245	TDno
226	DSTDxax	246	TDSxo
227	TSDTaoxn	247	TDSano
228	SDTSxax	248	TDSao
229	TDSTaoxn	249	TDSxno
230	SDTSanax	250	DTo
231	STxTDxan	251	DTSnoo
232	SSTxDSxax	252	TSo
233	DSTDSanaxxn	253	TSDnoo
234	DTSao	254	DTSoo
235	DTSxno	255	1

Pixel Placement

HP PCL 5 printers place pixels at the intersection of the squares of a theoretical, device-dependent grid covering the printable area on the page. Depending on the image and the logical operation in effect, a problem may occur when the sides of two polygons touch each other—the pixels along the common border may be printed twice or not at all. For example, a source rectangle consisting of all 1's that is XORed with a destination consisting of all 1's produces a white rectangle; but if another source rectangle is placed on the page touching the first rectangle, the two rectangles will be white-filled except at their common border ((1^1) ^ 1 = 1).

To correct situations where this problem occurs, the PCL printer language provides a choice of pixel placement models: grid intersection and grid centered. The grid intersection model is the default: pixels are rendered on the intersections of the device-dependent grid covering the page. In the grid-centered model, the number of rows and columns are each reduced by one, and pixels are placed in the center of the squares, rather than at the intersections.

The following example illustrates the concepts of the two models (see Figure 5-5). Assume a rectangle extends from coordinate position (1,1) to position (3,4). As shown below, for the same coordinates, the grid-centered model produces a rectangle that is one dot row thinner and one dot row shorter than the grid intersection model. Thus, the grid-centered model should be selected when two or more polygons on a page may share a common border.

Since PCL printers print only at the intersections of the grid, the actual implementation of the grid-centered model is shown on the right.

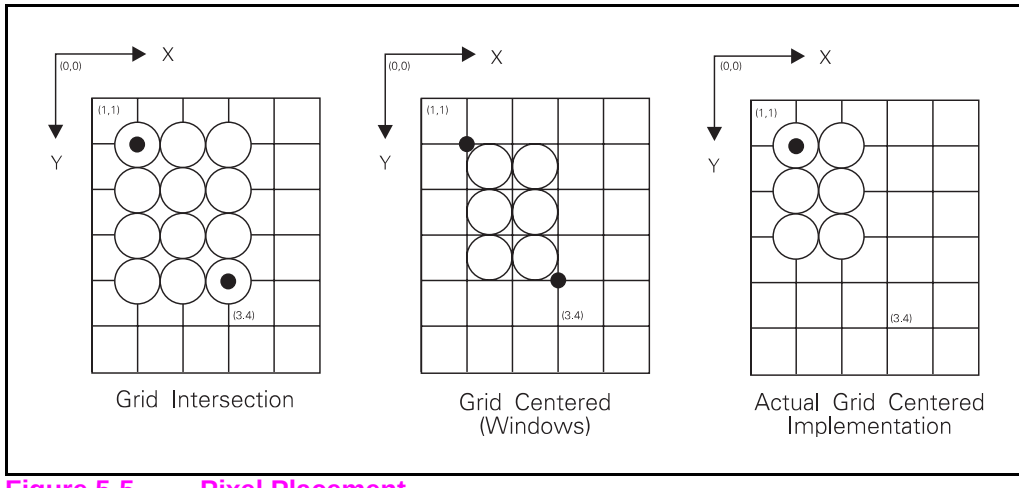

Figure 5-5 Pixel Placement

Note The grid-centered method is used by Microsoft Windows.

When rectangular area fills are used and grid intersection is used, an overlapping of pixels can occur if rectangular area fills are placed adjacent to one another (as shown below). Depending on the raster operation presently in effect, this overlap can produce undesirable results in the final printed image. To avoid this problem, use the grid-centered method.

Note Since PCL printers print only at intersections, grid- centered pixel placement is implemented as shown on the right.

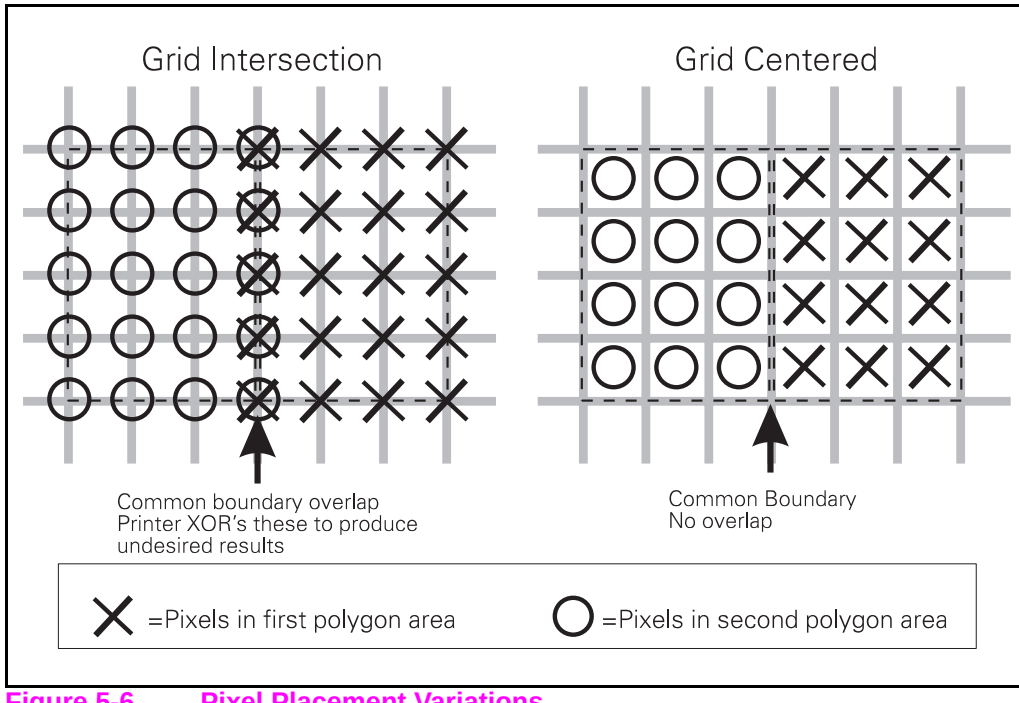

Figure 5-6 Pixel Placement Variations

There are two commands that modify the pixel placement function: the PCL Pixel Placement command (Ec*ℓ#R) and the HP-GL/2 Pixel Placement command (PP).

Pixel Placement Command

Determines how pixels are rendered in images.

$^{E}_{C}$*** ℓ # R**

# =	0	-	Grid intersection
	1	-	Grid centered

Default	=	0
Range	=	0, 1 (command is ignored for other values)

Two models are used for rendering pixels when an image is placed on paper:

- Grid Intersection Model
- Grid Centered Model

This command can be used multiple times per page. It has no effect except to switch the model being used for imaging.

Note

The PCL Pixel Placement command determines how pixels are placed for both PCL and HP-GL/2 operation.

This command performs the same function as the HP-GL/2 PP command described in Chapter 7.

Filling with Patterns

The procedure for applying patterns to text, raster images, and rectangular areas is essentially the same, except that for text and raster images the Current Pattern (EC*v#T) command is used, and for rectangular areas the Fill Rectangular Area (EC*c#P) command is used. The procedures below describe how to fill with PCL and HP-GL/2 patterns.

Patterns for Text and Raster Images

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

1. Specify the Pattern ID (EC*c#G) command. For HP-defined patterns, select an ID that specifies the desired pattern.
2. Download the pattern (EC*c#W). This step is for user-defined patterns only. The downloaded pattern adopts the current pattern ID.
3. Apply the pattern to all subsequent text and raster images. Specify the current pattern type (EC*v#T).

Patterns for Rectangles

Use the following general procedure to apply a non-solid pattern to rectangular areas.

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

HP-GL/2 Patterns

PCL patterns can be used in HP-GL/2 mode, but HP-GL/2 patterns cannot be used in PCL mode. Using HP-GL/2, patterns are downloaded using the RF (Raster Fill) command, and applied using the FT (Fill Type) or SV (Screened Vectors) commands.

Pattern ID (Area Fill ID) Command

The Pattern ID command (formerly called Area Fill ID) identifies the specific shading, cross-hatch, or user-defined pattern. (This command is also used for rectangular area fill, described later in this chapter.)

$^{E}_{C}$ * **c # G**

Selecting Shaded patterns:		Selecting Cross-Hatch patterns:	
# =	1 thru 2 = 1- 2% shade	# =	1 - Pattern #1
	3 thru 10 = 3-10% shade		2 - Pattern #2
	11 thru 20 = 11-20% shade		3 - Pattern #3
	21 thru 35 = 21-35% shade		4 - Pattern #4
	36 thru 55 = 36-55% shade		5 - Pattern #5
	56 thru 80 = 56-80% shade		6 - Pattern #6
	81 thru 99 = 81-99% shade		
	100 = 100% shade		

Selecting User-Defined patterns:[1]

= ID number of user-defined pattern

[1] Not supported on all PCL 5 printers. Refer to the “PCL Feature Support Matrix” in Chapter 1 of the *PCL 5 Comparison Guide* for specifics.

Default = 0 (no pattern)
Range = 0 – 32767 (values outside the range are ignored)

For rectangular areas, the pattern “material” is determined by both the pattern ID and the value of the Fill Rectangular Area command. For other images, the pattern material is determined by the pattern ID and the value of the Select Pattern command.

Figure 5-7 and Figure 5-8 illustrate the HP-defined shading patterns and cross-hatched patterns, respectively.

Note

This command is used for both the Select Pattern and Rectangular Area Fill graphics.

For user-defined patterns, this command, sent prior to downloading a user-defined pattern, assigns an ID pattern number to the downloaded pattern. (For more information, see “User-Defined Pattern Graphics,” later in this chapter.)

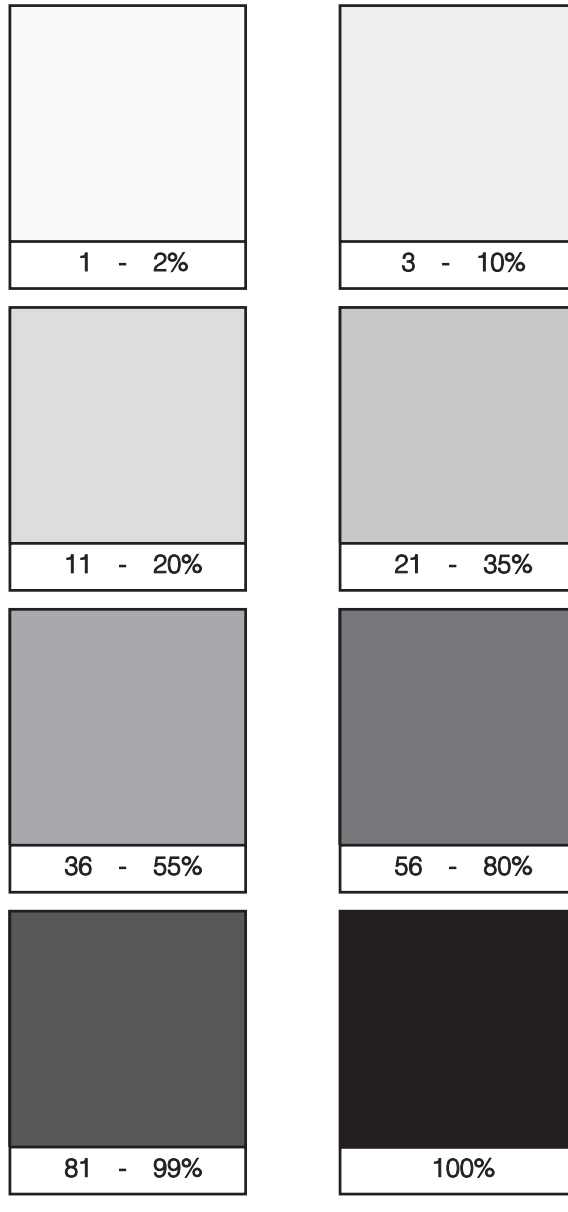

Figure 5-7 Shading Patterns

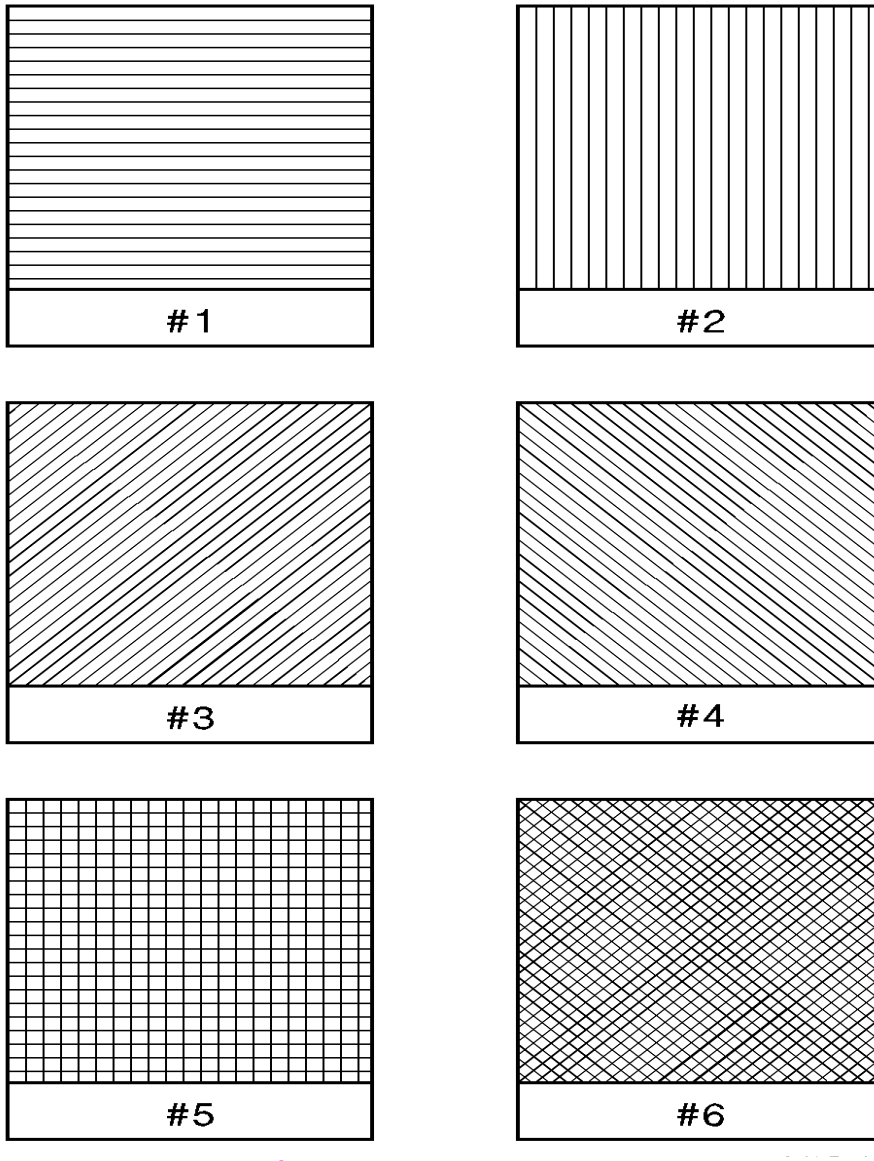

Figure 5-8 Cross-Hatch Patterns

Select Current Pattern Command

The Select Current Pattern command identifies the type of pattern to be applied onto the destination.

$^{E}_{C}$ * **v # T**

# = 0	-	Solid black or foreground color
1	-	Solid white
2	-	Shading pattern
3	-	Cross-hatch pattern
4	-	User-defined pattern

Default	=	0
Range	=	0 - 4 (values outside of range are ignored)

This command selects which type of pattern is applied. For values 2, 3, and 4, the desired shading level, cross-hatch pattern, or user-defined pattern number is identified by the Pattern ID command described earlier in this chapter.

Note

For selecting or changing the current pattern, the Select Current Pattern ($^{E}_{C}$*v#T) and the Pattern ID ($^{E}_{C}$*c#G) commands work together. **Sending the current pattern** (Select Current Pattern command) **alone does not change the current pattern; the Pattern ID must be sent first.** However, when selecting solid white (white rule) or solid black (black rule), only the Select Current Pattern command is required.

Once a current pattern is selected, that pattern applies to all images placed on the page until a new pattern is selected.

User-Defined Pattern Graphics

In addition to the eight shading patterns and six cross-hatch patterns, users can design their own fill patterns. These **user-defined patterns** are downloaded to the printer and controlled using three commands:

- Download Pattern ^{E}C*c#W [data]
- Set Pattern Reference Point ^{E}C*p#R
- Pattern Control ^{E}C*p#Q

Using User-Defined Patterns

To create a new pattern, a user defines a binary raster data image as a base pattern. This base pattern is downloaded to the printer using the User-Defined Pattern command. Prior to downloading the pattern, a Pattern ID command is sent to assign the user pattern an ID number. This ID number is used to select the pattern for printing and for pattern management.

To apply the pattern to an image, the printer duplicates or tiles (like placing ceramic tiles) the pattern across and down the page. This pattern can be applied to any image, including rectangular area fill.

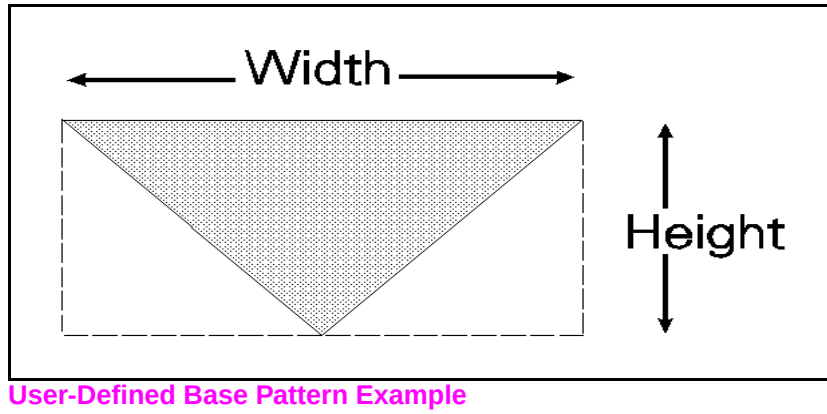

Figure 5-9 User-Defined Base Pattern Example

A user-defined pattern may be applied to any image in the same manner as the internal cross-hatch or shade patterns.

Note

For efficient memory usage and improved performance, it is strongly recommended that user-defined patterns should be 8x8, 16x16, or 32x32 in size. Specification of patterns that are either 1 pixel in height or width is strongly discouraged.

If user-defined halftones are also used, they need to be either the same size or multiples of each other to avoid render anomalies due to each pattern being rendered differently across the page (if tiled), or due to variations
in xy position.

How the Printer Tiles a Pattern

A user-defined base pattern is a rectangular binary pattern stored in the printer. To apply the pattern to an image area on the page, the printer duplicates the base pattern across and down the page. This process is referred to as tiling. (The pattern is only applied to those areas on the page for which the pattern is required.)

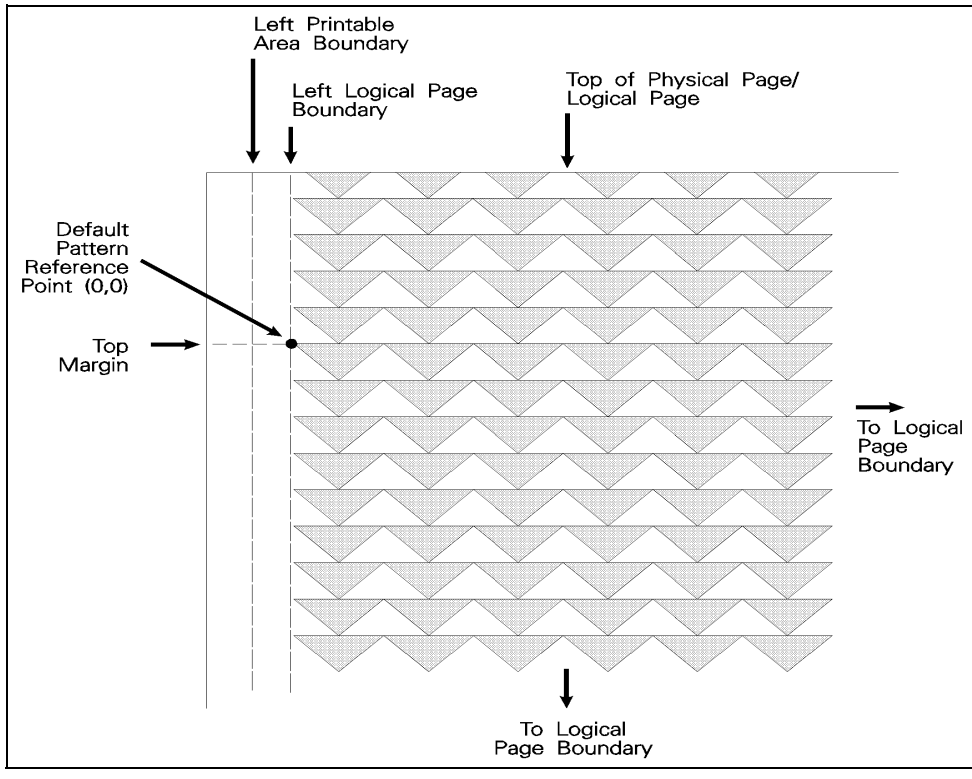

Figure 5-10 Pattern Layout Across the Printable Area

Pattern Reference Point

The pattern reference point is a position on the logical page at which the base pattern is positioned for tiling. The upper left corner of the base pattern is positioned at this point (see Figure 5-10). The default pattern reference point is position 0,0. However, it is possible to set the pattern reference point to the current cursor position. This allows the pattern to be positioned or adjusted for fill areas. The pattern reference point may be shifted more than once for as many fill areas as there are on a page (the area must be filled before the tile point is moved for the next fill area).

Figure 5-11 shows two areas filled with the pattern reference point fixed at the default (0,0) position. The lower portion of the illustration shows two areas in which the pattern reference point was moved to the upper left corner of each area and the area filled separately.

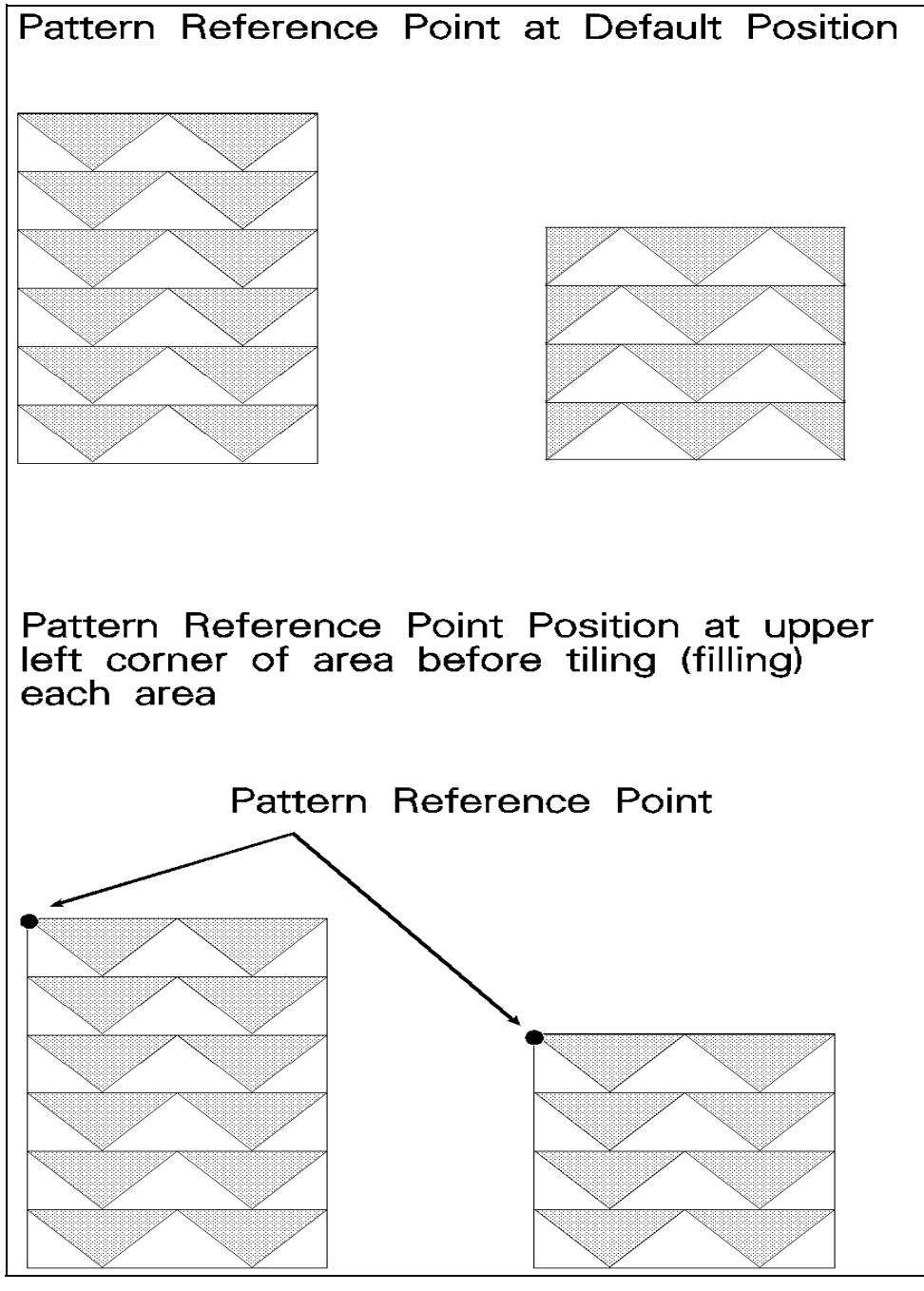

Figure 5-11 Moving Pattern Reference Point for Pattern Filling

Download Pattern Command

The Download Pattern command provides the means for downloading the binary pattern data that defines the user pattern.

$^{E}_{C}$ * **c # W [pattern data]**

= Number of pattern data bytes

Default = 0
Range = 0 – 32767 (HP Color LaserJet 8500 extends the range to 0 to 65535; values outside the range are ignored)

The value field (#) identifies the number of pattern data bytes that follow the Download Pattern command. In addition to the binary pattern data, there are eight bytes of pattern descriptor (header) information included in this pattern data. The format for a 300 dpi resolution header is shown in Table 5-5, below.

Table 5-5. User-Defined Pattern Header (300 dpi resolution)

Byte	**15 - MSB 8**	**7 LSB-0**	Byte
0	Format (0)	Continuation (0)	1
2	Pixel Encoding (1)	Reserved (0)	3
4	Height in Pixels		5
6	Width in Pixels		7
8	Pattern image		

Format (Byte 0)

This field indicates the downloadable pattern format:

Format 0	1 bit per pixel: black-and-white or foreground color. A “1” bit indicates black or foreground color for a color pattern. A “0” indicates either white or transparency, depending on the source and pattern transparency modes. A “0” cannot be colored.
Format 1	1 or 8 bits per pixel. This format uses 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.

Continuation (Byte 1)

This field, byte 1, must be set to “0.” (This byte is for future printer support and does not currently provide any continuation operation.)

Pixel Encoding (Byte 2)

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 two-color palette, 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.

7 5	4	3 0
000	Unused	Bits/Index

Reserved (Byte 3)

This field, byte 3, is not currently used and must be set to 0.

Height in Pixels (Bytes 4 and 5)

This field, bytes 4 and 5, identifies the number of raster rows (height) of the pattern, specified at device resolution. If the height is 0, the data is ignored and no pattern is defined. Pattern height must be less than 32767 pixels.

Width in Pixels (Bytes 6 and 7)

This field, bytes 6 and 7, identifies the number of pixels (width) of the pattern, specified at device resolution. If the width is 0, the data is ignored and no pattern is defined. Pattern width must be less than 32767 pixels.

Pattern Image

This field contains the raster data for the pattern. Data rows must be word-aligned. Pattern image data is formatted differently for each format type (see the data description under “Format (Byte 0)” on the previous page).

User-defined Pattern Example

This example shows how the user-defined pattern command is used to create new patterns. For this example, a pattern of triangles is used. The first step is to design the base pattern triangle (in this case, using 64 bytes of data). The base pattern binary data is shown below:

```
11111111111111111111111111111111
01111111111111111111111111111110
00111111111111111111111111111100
00011111111111111111111111111000
00001111111111111111111111110000
00000111111111111111111111100000
00000011111111111111111111000000
00000001111111111111111110000000
00000000111111111111111100000000
00000000011111111111111000000000
00000000001111111111110000000000
00000000000111111111100000000000
00000000000011111111000000000000
00000000000001111110000000000000
00000000000000111100000000000000
00000000000000011000000000000000
```

This translates into the following 64 bytes of hexadecimal values:

```
FF  FF  FF  FF
7F  FF  FF  FE
3F  FF  FF  FC
1F  FF  FF  F8
0F  FF  FF  F0
07  FF  FF  E0
03  FF  FF  C0
01  FF  FF  80
00  FF  FF  00
00  7F  FE  00
00  3F  FC  00
00  1F  F8  00
00  0F  F0  00
00  07  E0  00
00  03  C0  00
00  01  80  00
```

When using the 300 dpi User-Defined Pattern header (see Table 5-5), set the eight bytes of header information to the following values:

Byte 0 – Format 0 (00 hex)

Byte 1 – Continuation 0 (00 hex)

Byte 2 – Pixel Encoding 1 (01 hex)

Byte 3 – Reserved 0 (00 hex)

Byte 4/5 – Height in Pixels 0 / 16 (00 / 10 hex)

Byte 6/7 – Width in Pixels 0 / 32 (00 / 20 hex)

Byte 8 – Begins the first bytes of binary data.

The PCL code below downloads the user-defined pattern and assigns it an ID number of 3.

1. Specify the pattern ID number:

^{E}C*c3G	Assigns an ID number of 3 to the pattern data which follows.

2. Send the User-defined Pattern command:

^{E}C*c72W	Specifies that 72 bytes are to follow (8 bytes for the header plus 64 bytes of pattern data).

Send the pattern header and binary data:

```
00 00 01 00 00 10 00 20
FF FF FF FF
7F FF FF FE
3F FF FF FC
1F FF FF F8
0F FF FF F0
07 FF FF E0
03 FF FF C0
01 FF FF 80
00 FF FF 00
00 7F FE 00
00 3F FC 00
00 1F F8 00
00 0F F0 00
00 07 E0 00
00 03 C0 00
00 01 80 00
```

Note

There must be an even number of bytes in user-defined pattern data, hence the trailing zeros (“padding”) in the last eight data rows above.

In the previous example, the raster data code is presented in hexadecimal, however, the numbers in the escape sequences are decimal.

Set Pattern Reference Point Command

The Set Pattern Reference Point command causes the printer to tile patterns with respect to the current cursor position (CAP). This command also specifies whether the pattern rotates with the print direction or remains fixed.

$^{E}_{C}$ * **p # R**

# =	0	-	Rotate patterns with print direction
	1	-	Keep patterns fixed

Default	=	0
Range	=	0,1 (values outside the range are ignored)

A value field of 0 rotates the patterns with changes in the print direction (see Print Direction command). For a value field of 1, patterns remain fixed for changes in print direction.

The default pattern reference point is the upper left corner of the logical page at the top margin (position 0,0). If the Set Pattern Reference Point command is not set, the pattern is tiled with respect to the default reference point.

Note

All patterns are rotated for changes in orientation, but the pattern reference point remains the same (refer to “Logical Page Orientation Command” in Chapter 5 of the *PCL 5 Printer Language Technical Reference Manual*).

This command applies to user-defined, shading, and cross-hatch patterns.

Pattern Control Command

The Pattern Control command provides a means for manipulating user-defined patterns.

$^{E}_{C}$ * **c # Q**

# =	0	-	Delete all patterns (temporary & permanent)
	1	-	Delete all temporary patterns
	2	-	Delete pattern (last ID # specified)
	4	-	Make pattern temporary (last ID # specified)
	5	-	Make pattern permanent (last ID # specified)

Default	=	0
Range	=	0, 1, 2, 4, 5 (command is ignored for other values)

For value fields 2, 4, and 5, the Pattern ID ($^{E}_{C}$*c#G) command is sent prior to the Pattern Control command to identify the specific pattern to which the Pattern Control command action is applied.

Rectangular Area Fills (Rules)

Rectangular area fills are a special case of source images—the source transparency mode has no effect, since the printer treats the rectangular area as a solid “black” (all 1’s) source.

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

Filling a rectangular area does not change the current active cursor position (CAP). The filled rectangular area is not affected by end-of-line wrap, perforation skip mode, or margins. A rectangular area may extend beyond the margins, but it will be clipped to the printable area of the logical page. Rectangular areas are not affected by graphics resolution (Ec*t#R).

Except for the absence of white pixels in the source, pattern transparency operates the same way for rectangular area fills as for other sources. The non-white pixels of the pattern are poured through the entire rectangular area onto the destination. The white bits of the pattern are either applied or ignored, based on the pattern transparency mode. If foreground color is used, it is applied to the non-white bits of the pattern prior to pouring (except for user-defined color patterns).

Note The Pixel Placement command (Ec*ℓ#R) affects rules.

The commands used to print rectangular area fills are described beginning on the next page.

Horizontal Rectangle Size (PCL Units)

This command specifies the horizontal rectangle size in PCL Units.

$^{E}_{C}$ * **c # A**

= number of PCL Units (valid to 4 decimal places)

The horizontal rectangle size is clipped to the bounds of the logical page. Values greater than the logical page boundary are acceptable; however, the final output is limited to the printable area of the logical page. Values outside the range of 0 – 32767 are ignored.

The default rectangle size is 0. Power-up and reset return this value to the default.

Horizontal Rectangle Size (Decipoints)

This command specifies the horizontal rectangle size in decipoints.

$^{E}_{C}$ * **c # H**

= number of decipoints (valid to 4 decimal places)

The horizontal rectangle size is clipped to the bounds of the logical page. Values greater than the logical page boundary are acceptable; however, the final output is limited to the printable area of the logical page. Values outside the range of 0 – 32767 are ignored.

Decipoints are converted into printer dot values, and any fraction of a dot is rounded up to the next full dot size.

The default rectangle size is 0. Power-up and reset return this value to the default.

Vertical Rectangle Size (PCL Units)

This command specifies the vertical rectangle size in PCL Units.

$^{E}_{C}$ * **c # B**

= number of PCL Units (valid to 4 decimal places)

The vertical rectangle size is clipped to the bounds of the logical page. Values greater than the logical page boundary are acceptable; however, the final output is limited to the printable area of the logical page. Values outside the range of 0 – 32767 are ignored.

The default rectangle size is 0. Power-up and reset return this value to the default.

Vertical Rectangle Size (Decipoints)

This command specifies the vertical rectangle size in decipoints.

$^{E}_{C}$ * **c # V**

= number of decipoints (valid to 4 decimal places)

The vertical rectangle size is clipped to the bounds of the logical page. Values greater than the logical page boundary are acceptable; however, the final output is limited to the printable area of the logical page. Values outside the range of 0 – 32767 are ignored.

Decipoints are converted into printer dot values, and any fraction of a dot is rounded up to the next full printable dot.

The default rectangle size is 0. Power-up and reset return this value to the default.

Fill Rectangular Area

The Fill Rectangular Area command determines the type of pattern used to fill the rectangle.

$^{E}_{C}$ * **c # P**

# =	0	-	Solid black or foreground color
	1	-	Solid white fill
	2	-	Shaded fill
	3	-	Cross-hatch fill
	4	-	User-defined pattern fill
	5	-	Current pattern fill

Default = 0
Range = 0 - 5 (out-of-range values are ignored)

Note

If a foreground color is selected, solid, shaded, and cross-hatch patterns are printed in the foreground color. User-defined patterns are not affected by the foreground color, but can contain color if they are defined as such. Solid white fills are not affected by foreground color.

Black fill—fills the rectangular area with black fill or with the current foreground color.

White fill—erases any fill in the rectangular area (it fills the rectangular area with white fill). Pertaining to white fills, the pattern transparency mode is always “opaque” (that is, the white pixels always have an effect on the destination).

Shaded fill—fills the rectangular area with one of eight shading patterns as specified by the Pattern ID command.

Cross-Hatch fill—fills the rectangular area with one of the six cross-hatched patterns as specified by the Pattern ID command.

User-defined fill—fills the rectangular area with custom pattern data as specified by the Pattern ID command and downloaded by the User-Defined Pattern command.

Current Pattern—fills the rectangular area with the current pattern.

Note

The current pattern is not applied to a rectangular area unless specified by this command.

The order in which data (patterns/rules, text, raster) is received is the order in which it is processed during the rasterization of the page.

The fill or pattern used as the current pattern is selected using the Select Current Pattern (^{E}C*v#T) command.

Black fill (value field 0), also known as black rule, and the white fill (value field of 1) “patterns” do not have a choice of different patterns, and thus do not require a pattern specification using the Pattern ID command.

The upper left corner of the rectangular area is located at the cursor position when printing a rectangular area. After printing the rectangular area the cursor is returned to the upper left corner; the cursor position does not change positions as a result of printing a rectangular area.

Rectangular areas are independent of the text area and perforation skip mode; these boundaries are ignored (rectangles are not clipped at these boundaries). Addressable rectangular areas are limited to the logical page. Rectangular areas that extend outside the logical page are clipped at the logical page boundaries (refer to the *PCL 5 Printer Language Technical Reference Manual* for logical page and printable area boundary specifications).

The pattern transparency mode controls how the area fill pattern is applied to the page. Refer to the following section for a description of how the pattern transparency mode affects the rectangular fill area.

A white fill “erases” any data placed within the rectangular area, regardless of the transparency mode settings. However, after a white fill erases data within an area, data subsequently placed within that area will be visible.

Pattern Transparency for Rectangular Area Fill

Pattern transparency affects how a pattern is applied to the rectangular fill area. The pattern and pattern type are selected by the Pattern ID command (EC*c#G) and the Fill Rectangular Area (EC*c#P) command (described earlier in this chapter).

Note Source transparency has **no effect** on the rectangular fill area since the rectangular area is viewed as all 1's (a solid black source image).

When applying a pattern (area fill) to the rectangular area, the pattern transparency mode affects the final result the same as it does when filling other images or text. The pattern transparency mode determines the effect white pixels of the pattern have on the destination for value fields 0 (black fill), 2 (shaded fill), 3 (cross-hatch fill), or 5 (current pattern fill) of the Fill Rectangular Area command.

The "0'' bits of the fill pattern are either applied (opaque) or ignored (transparent) based on the transparency mode setting (see Figure 5-12). When a value field of 1 (white fill) is used, pattern transparency mode is **always treated as if it were opaque**.

The effect of transparency modes on rectangular areas is illustrated in Figure 5-12. In both examples, the source transparency mode is opaque regardless of the actual setting. In the first example, the pattern transparency mode is transparent; the white pixels in the pattern are not applied to the destination, so that the pattern is visible in only two quadrants of the destination. In the second example, the pattern transparency mode is opaque, and the pattern is visible in the entire rectangular area.

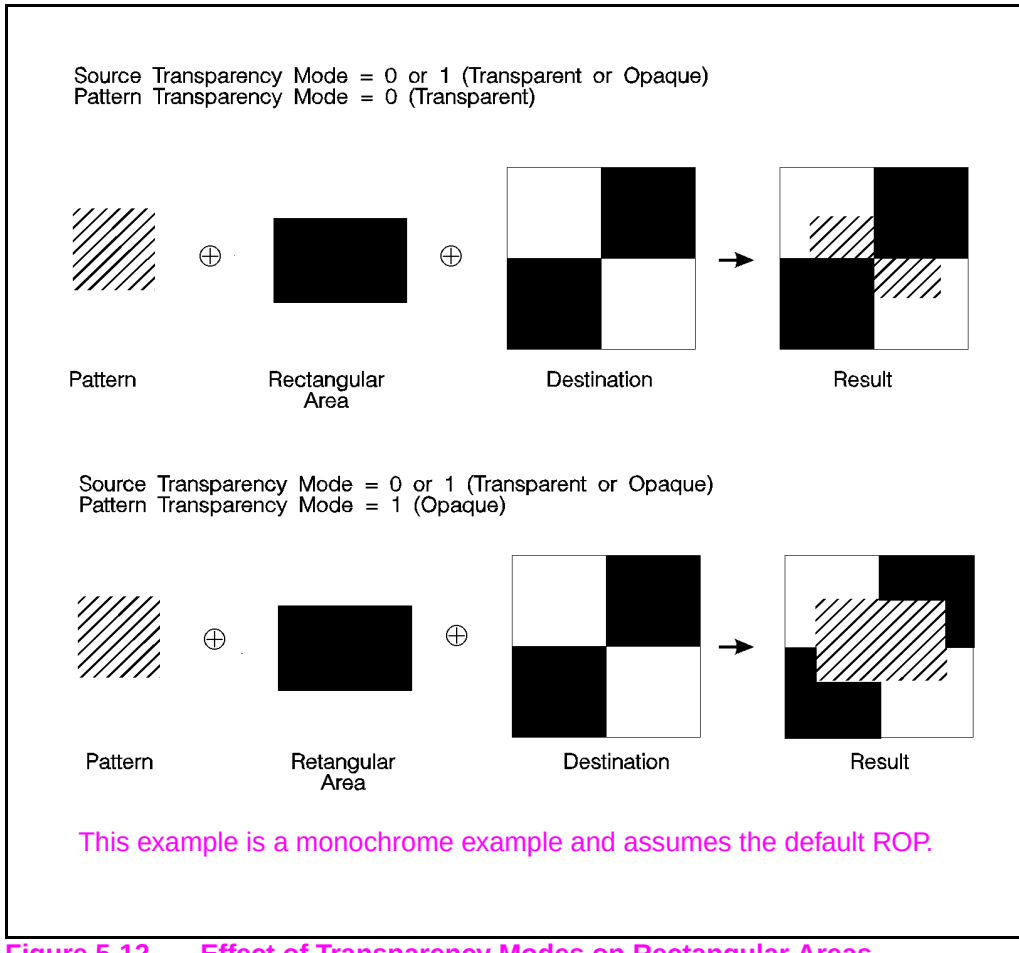

Figure 5-12 Effect of Transparency Modes on Rectangular Areas

Rectangular Fill Examples

Following are two examples that demonstrate the way to print and fill rectangular shapes. The first example demonstrates filling rectangles with solid fill and the second example demonstrates filling with a shading pattern.

Solid Fill (Black/White)

To print a 900 by 1500 Unit black rule (3 inches by 5 inches at 300 units-per-inch), then “white fill” a small area inside the black rectangle, perform the following steps.

1 Position the cursor:

^EC*****p300x400Y	This moves the cursor to PCL Unit position (300, 400) within the PCL coordinate system.

2 Specify the width of the rule:

^EC*****c900A	This sets the rule width to 900 PCL Units (3 inches at 300 units-per-inch).

3 Specify the height of the rule:

^EC*c1500B	This sets the rule height to 1500 PCL Units (5 inches at 300 units-per-inch).

4 Print the rule:

^EC*****c0P	This example prints a black filled rectangular area.

5 Position the cursor inside the rectangular area:

^EC*****p600x700Y

6 Specify the width and height for the smaller white fill rectangular area:

^EC*****c300a600B

7 Select the white fill and print.

$^{E}_{C}$*c1P

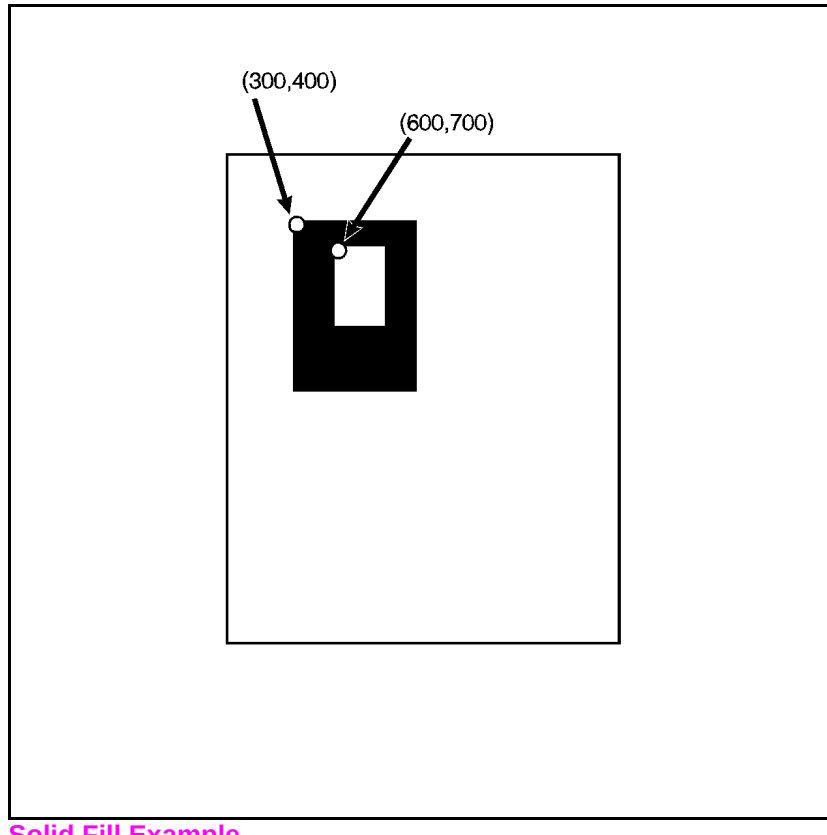

Figure 5-13 Solid Fill Example

Shaded Fill

To print a 900 by 1500 Unit 25% shaded rectangle (3 inches by 5 inches at 300 units-per-inch), perform the following steps.

1 Position the cursor:

E_C*****p300x400Y	This moves the cursor to PCL Unit position (300, 400) within the PCL coordinate system.

2 Specify the width of the rectangle:

E_C*****c900A	This sets the rectangle width to 900 PCL Units (3 inches at 300 units-per-inch).

3 Specify the height of the rectangle:

E_C*****c1500B	This sets the rectangle to 1500 PCL Units (5 inches at 300 units-per-inch).

4 Specify the Pattern ID:

E_C*****c25G	This sets the Pattern ID to 25.

5 Print the rectangular shaded area:

E_C*****c2P This example prints the following:

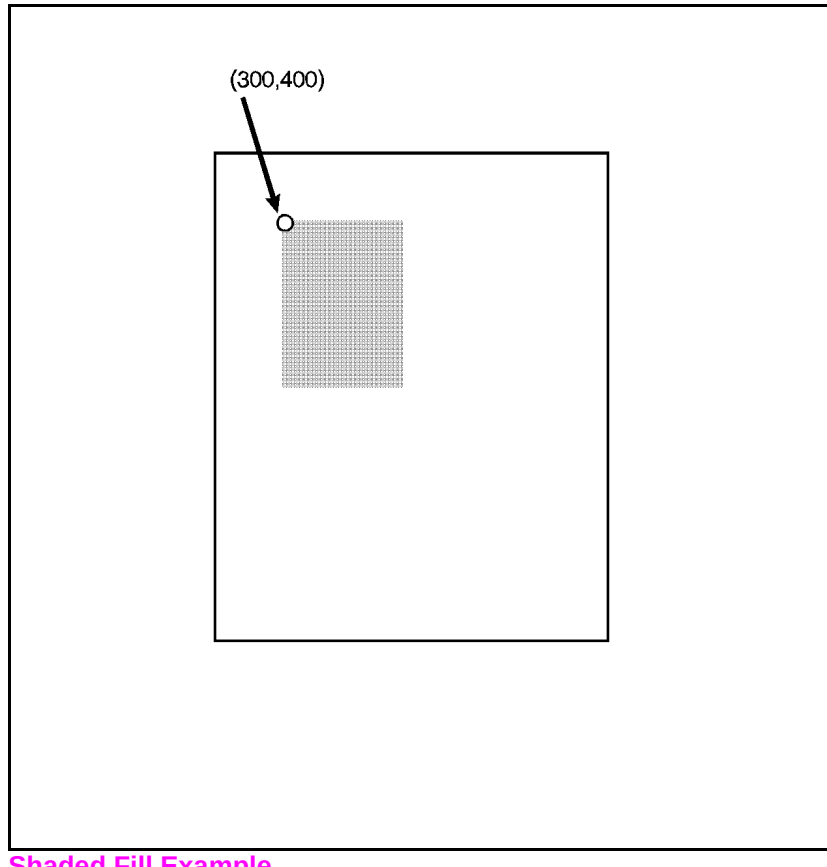

Figure 5-14 Shaded Fill Example