Chapter 14: Color

Contents of this Chapter

This chapter describes the following PCL color commands:

Assign Color Index	*Esc*v#I*	14-36
Color Component 1	*Esc*v#A*	14-36
Color Component 2	*Esc*v#B*	14-36
Color Component 3	*Esc*v#C*	14-36
Color Lookup Tables	*Esc*l#W*	14-48
Configure Image Data (CID)	*Esc*v#W*	14-15
Download Dither Matrix	*Esc*m#W*	14-44
Finish Mode	*Esc*o#A*	14-51
Foreground Color	*Esc*v#S*	14-41
Gamma Correction	*Esc*t#I*	14-49
Monochrome Print Mode	*Esc&l#B*	14-51
Palette Configuration	*Esc&p#C*	14-32
Palette Control	*Esc&p#C*	14-40
Palette Control ID	*Esc&p#I*	14-39
Push/Pop Palette	*Esc*p#P*	14-37
Render Algorithm	*Esc*t#J*	14-42
Select Palette	*Esc&p#S*	14-39
Simple Color	*Esc*r#U*	14-13
Viewing Illuminant	*Esc*i#W*	14-50

14.1 Introduction

Except for HP-GL/2 vector graphics, PCL color uses the raster commands described in Chapter 13 in addition to those contained in this chapter. The following additional definitions are necessary.

Palette

A palette is a collection of colors that are selected by index numbers. All PCL color modes create default palettes. Palettes can be pushed onto a stack and later popped. Only one palette can be active at a time. The active palette is overwritten whenever a palette is created or popped from the stack. The active palette is transferred between PCL and HP-GL/2. The palette is always used to select non-raster colors, but not always for raster colors.

Raster Color vs. Non-Raster Color

Palettes are used differently depending on whether the printer is in raster mode.

- In non-raster mode, the palette is always used to select colors. The color of text or patterns is specified by the Foreground Color command (*Esc*v#S*).
- In raster mode, the palette is used for *indexed* but not *direct* color selection (see the Color Selection section below).

Device-Independence vs. Device-Dependence

Device-independent color is specified absolutely, in a coordinate system that is independent of any device. A red specified in a device-independent color space will look the same on any two printers, even if they must combine different amounts of cyan, magenta, yellow, and black ink. In order to produce device-independent color, a printer must be calibrated to a color standard.

Device-dependent color spaces are relative to the device's native rendering capability. A red specified in a device-dependent color space may look different on any two printers — even if they combine exactly the same amounts of cyan, magenta, yellow and black — because of the different properties of the ink or toner.

Black and White References

Device-dependent specifications use an arbitrary range of values for each primary component. Range endpoints, or *black and white references*, are specified for each component. A color is derived by specifying the value of each of its components relative to these predefined limits.

For Device RGB color space, the white reference represents the maximum output of a primary that a device can produce; and the black reference represents the minimum output of that primary. For example, if 100 is chosen as the white reference for red in the RGB color model, it represents the reddest red the device can produce. If 10 were chosen instead, then 10 would represent the same red.

For example, if the red, green and blue white references are set to 63 and the black references are set to 0 (white = 63, 63, 63 and black = 0, 0, 0), then 50% blue = 0, 0, 31. But if the white and black references are set to white = 63, 127, 31 and black = 4, 0, 0, then 50% blue = 0, 0, 15.

Color Selection

Colors are selected by two methods. In ***indexed*** selection, a color is selected by its index number in a palette. In ***direct*** selection, colors are specified by the proportions of their primary components. Raster mode may use either type of selection; non-raster mode uses only indexed selection.

Indexed

Indexed selection specifies a color by its palette index number. In non-raster mode, the Foreground Color command (*Esc*v#*S) selects a color by palette index. In raster mode, the bit combination for each pixel forms an index number. For example, three bits can specify any index in an 8-color palette:

Three-bit combinations	0 1 0 1 0 1 0 1	lsb
	0 0 1 1 0 0 1 1	
	0 0 0 0 1 1 1 1	msb
Palette index number	0 1 2 3 4 5 6 7	

Direct

Direct color selection specifies a color by the proportions of its primary components. For example, a 24-bit-per-pixel representation may be (0xff,0xf0,0x00) for red, green, and blue (a slightly red-tinted yellow). Non-raster colors cannot be selected directly. Raster colors may be selected directly or indirectly, depending on the meaning of the bits that are transferred; that is, the transmitted bit combinations for each pixel may form a palette index number, or they may directly specify component proportions. Byte #1 of the Configure Image Data command (*Esc*v#W*) determines the format in which raster data is to be transmitted and interpreted. Palette entries may be reprogrammed by direct specification (*Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, or *PC*).

Data Encoding

When sending raster data to the printer, several bits are needed to specify a pixel's color. The pixels in a row of raster data may be encoded using either *plane* or *pixel* format.

Planar encoding sends one bit for each pixel in a row. This partial specification for all the pixels in a row is called a plane. Subsequent planes are sent until all the pixels in the row are completely specified.

Pixel encoding sends all the bits for each pixel before sending the bits for the next pixel.

Encoding by Plane

Planar encoding uses successive data planes, each providing one bit for each pixel in a row. Each plane builds upon the preceding planes until the pixels in a row are fully defined. In the example below, the highlighted bits compose the color index for the third pixel in the first row.

*Esc*b#V*	row 1	plane 1	b1 b1 **b1** b1 b1 b1 ...
*Esc*b#V*		plane 2	b2 b2 **b2** b2 b2 b2 ...
*Esc*b#W*		plane 3	b3 b3 **b3** b3 b3 b3 ...
*Esc*b#V*	row 2	plane 1	b1 b1 b1 b1 b1 b1 ...

Encoding by Pixel

In pixel encoding, each pixel is fully specified before any bits are sent for the next pixel. For example, if four bits are needed to define a pixel, then every group of four bits in the data stream defines a pixel. The highlighted (c4...c1) group below defines the second pixel in the first row.

*Esc*b#W*	row 1:	b4 b3 b2 b1 **c4 c3 c2 c1** ...
	row 2:	b4 b3 b2 b1 ...

The table below shows the PCL options for selecting colors and encoding color raster data.

	Planar Encoding	**Pixel Encoding**
Indexed Selection	Indexed planar	Indexed pixel
Direct Selection	Direct planar	Direct pixel

Color Modes

Four color modes are used in PCL:

- Black and White
- Simple Color
- PCL Imaging
- HP-GL/2 Imaging
- Configure Raster Data (see Chapter 15)

All of these modes create a palette; but palettes created in Black and White mode and Simple Color modes are not modifiable. Only one palette at a time can be active.

Black and White Mode (Default)

PCL devices power up in this mode and revert to it after an *EscE*. Black and White Mode creates an unmodifiable, default 2-pen palette, with white at index 0 and black at index 1.

Simple Color Mode

Simple Color (*Esc*r#U*) creates a fixed size, fixed color, unmodifiable palette. Palettes may be 2-pen black and white, 8-pen RGB, 8-pen CMY, or 16-pen KCMY. Data encoding is indexed planar.

PCL Imaging Mode

This mode is enabled by Configure Image Data (*Esc*v#W*) or Configure Raster Data (*Esc*g#W*). It allows a maximum of 24 bits per pixel for color specification. Therefore, more colors (produced by halftoning) may be specified than are available in Simple Color Mode. Pixel encoding mode, bits per pixel, bits per primary, white/black references, and the color palette are all programmable.

The Configure Raster Data command (*Esc*g#W*) described in Chapter 15 implements many raster configurations (e.g., different horizontal and vertical resolutions, multiple intensity levels, independent palette configuration, mixed indexed and monochrome, etc).

HP-GL/2 Imaging Mode

In HP-GL/2, the *IN* and *BP* commands start color imaging and perform the following:

- Set pixel encoding mode to index by plane.
- Set bits per index to 3.
- Create an 8-pen palette that is reprogrammable in either PCL or HP-GL/2.

Although default HP-GL/2 palettes are different than default PCL palettes, an HP-GL/2 palette is modifiable in either PCL or HP-GL/2. Also, a PCL palette created by the Configure Image Data command (*Esc*v#W*) is modifiable in HP-GL/2

The active palette is always transferred between HP-GL/2 and PCL contexts. Since only one palette at a time can be active, a new palette created in either context overwrites the current palette.

Device-Independent Color

The PCL language characterizes color rendering as either ***device-dependent*** or ***device-independent***. Both categories encompass many color spaces, each with unique characteristics.

DEVICE NOTE: DJs do not support device-independent color.

Device-Dependent Color

A device-dependent color specification is relative to the device's native rendering mode. Examples are:

- **Plotter:** The installed pens determine the color space. The color of a pen is never guaranteed.
- **Screen monitor:** The red, green, and blue screen phosphors determine the color space. Fully saturated colors vary between screens.
- **Printer:** The color produced on a page depends on the printer's subtractive inks or toner (cyan, magenta, yellow, and black).

Devices receiving relative color specifications for the same color may not produce the same color. A monitor's saturated red may be different from a plotter's. Different marking devices may produce different appearances for the same color page.

Device-independent Color

Device-independent color is based on the tristimulus values of human vision. An independent color specification is translated into a device's native space in such a way that the resultant color is independent of the device. Examples include Kodak PhotoYCC, CIE L*a*b*, YUV and proposed YCrCb. Each is a 3x3 linear transform from tristimulus XYZ space.

With proper calibration, any device can provide a transform from device-independent space into its native space; and different devices will produce output with the same color appearance. For example, if a monitor's parameters are known (gamma, gain, chromaticity coordinates for each primary, and white point), screen RGB pixel information can be transformed into device-independent color.

Color Matching

Proper device calibration can achieve ***true color matching***, wherein a side-by-side comparison of a printed page with the monitor on which the page was designed will show an exact match. However, in true color matching, printed colors are satisfactory only with the monitor as a viewing reference. Away from the screen context, the printed page appears flat and unsaturated because the printer and the monitor have different dynamic ranges. Screen black appears gray when compared to printed black; but this is unacceptable if the intent is pure black. Screen white appears yellow or blue when compared to a white sheet of paper, so true color matching would require that colored dots be printed in the white areas; and this is unacceptable if the intent is pure white.

Color Appearance Matching

Color ***appearance matching*** goes beyond true color matching by including the dynamic ranges of the devices, so user intent is maintained. Although printed color does not exactly match screen color, color appearance does match; and this is what the user really wants (studies show that participants like appearance matches better than true matches because they receive what they requested in terms of color quality). The PCL language uses appearance matching when rendering device-independent color to maximize user satisfaction.

Color Lookup Tables

Color lookup tables, which provide additional control of the printed output, are transformations that map input data into a new output range based upon point-by-point conversions.

Color transparencies provide one example of how to use color lookup tables. On plain paper, color appearance matching will satisfy expectations; but if the page is printed on transparency film, the resulting overhead image will be unsaturated and flat. To compensate by increasing color saturation without changing composition, the user can send a color lookup table in terms of CIE L*a*b* and increase a* and b* in equal amounts.

Color lookup tables could also be used to adjust data from a Kodak CD-ROM, which uses the PhotoYCC device-independent color space. The gamma correction table is complex and cannot be described by the traditional logarithmic expression. Since the data can be mapped into new data values via tables, the user can provide a gamma correction table that essentially describes the complex correction factors.

Color lookup tables can be used to "neutral-balance" an image. For example, an underwater photograph produces a severe bluish cast when printed. The user can eliminate that cast from the image by providing color lookup tables that subtract some color portion from each of the primaries.

Illumination Models

Illumination sources have different spectral distributions, reducing the perception of colors with spectral characteristics outside the illumination range. For example, printed colors shift in hue between fluorescent and tungsten lighting. PCL lets the user specify the x and y chromaticity values of a relative white point that can be used with a given illuminant.

Color Processing

PCL color graphic commands use a "color processing pipeline" for color generation. Although the pipeline is generalized below, it is not implementation specific; its purpose is to show where values are inserted into the overall flow. Although implementation details may vary, the diagrams on the following pages provide a conceptual view of how color is inserted and transformed in the system.

The PCL language uses color in two different ways:

- Page marking primitives
- Color raster

Page Marking Primitives

HP-GL/2 and PCL page marking primitives (e.g., glyphs, rules, polygons, circles, vectors) contain no information about the color of a picture. They merely mark the page with attributes assigned to the current working environment (e.g., colors, patterns, logical operation modes, etc.). They act like stencils through which the color paint is poured, forming a homogeneous pattern.

Color Raster

Unlike page marking primitives, each pixel of a color raster area contains information about the color of a given point.

A color raster pixel may be defined by either:

- Palette entry indices
- Direct color specifications

User-defined color patterns are a form of color raster; but each pixel of a user-defined color pattern can be defined only by palette entry indices, **not** by direct color specifications.

Color Processing Functions

Given these two color uses, color processing must:

- Convert color attributes to an internal representation that can be poured through the page marking stencil onto the destination via some logical operation.
- Convert multiple-bit-per-pixel color raster to an internal representation that can be merged into the destination via some logical operation.

Color processing must have access to the following state variables, which indicate the form and attributes by which the two color groups are generated.

- Halftone (rendering algorithm)
- RGB gamma correction
- Device-dependent color lookup tables for each of the three primaries

Device-dependent Color Spaces

The following PCL commands establish color processing for device-dependent color spaces:

Render Algorithm	*Esc*t#J*
Gamma Correction	*Esc*t#I*
Color Lookup Tables	*Esc*l#W*
Configure Image Data	*Esc*v#W*
Simple Color	*Esc*r#U*
Monochrome Print Mode	*Esc&b#M*

Color lookup tables or gamma correction (which are mutually exclusive) can modify the mapping of input to output.

Device-independent Color Spaces

The following PCL commands establish color processing for device-independent color spaces:

Render Algorithm	*Esc*t#J*
Gamma Correction	*Esc*t#I*
Color Lookup Tables	*Esc*l#W*
Configure Image Data	*Esc*v#W*
Viewing Illuminant	*Esc*i#W*
Monochrome Print Mode	*Esc&b#M*

Device-independent color spaces are supported under the following conditions:

1. The Configure Image Data command (*Esc*v#W*) configures the current palette and specifies a device-independent color space.

2. The Render Algorithm command (*Esc*t#J*) must be set to one of the following:

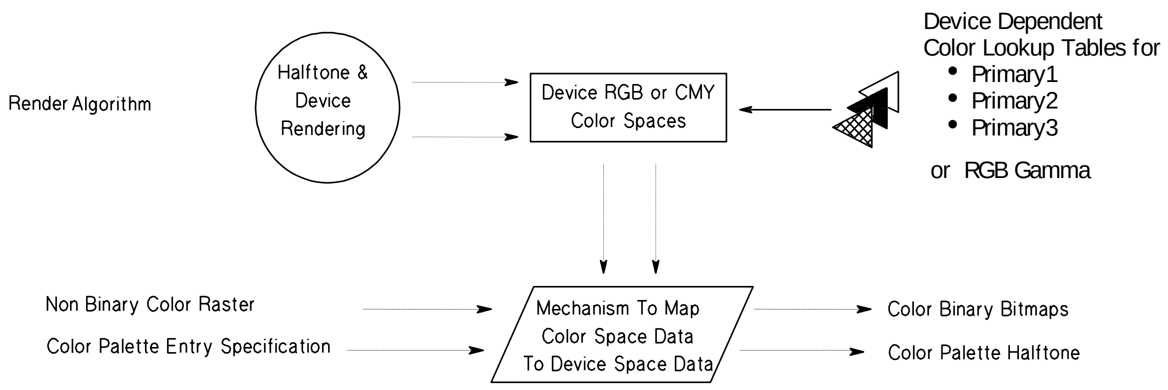
• Nearest Intensity	*Esc*t0J*
• Device Best	*Esc*t3J*
• Error Diffusion	*Esc*t4J*
• Cluster Ordered Dither	*Esc*t7J*
• Ordered Dither	*Esc*t11J*
• Noise Ordered Dither	*Esc*t10J*

Monochrome render algorithm selections are also allowed.

Color processing becomes device-dependent if the render algorithm is changed from one of the above. This is because extensive device characterization is necessary to achieve device-independence: calibration must be based on known parameters that affect the device's color gamut. Render algorithms such as Snap to Primaries (*Esc*t1J*), Snap Black to White and Colors to Black (*Esc*t2J*), or User-defined Halftone (*Esc*t9J*) either limit the number of colors available, or are undefined to the extent that their performance is not guaranteed; they therefore produce device-dependent results. Device-independent color is again generated if the render algorithm changes to one of the above and the color space has not changed.

Device-Dependent Pipeline

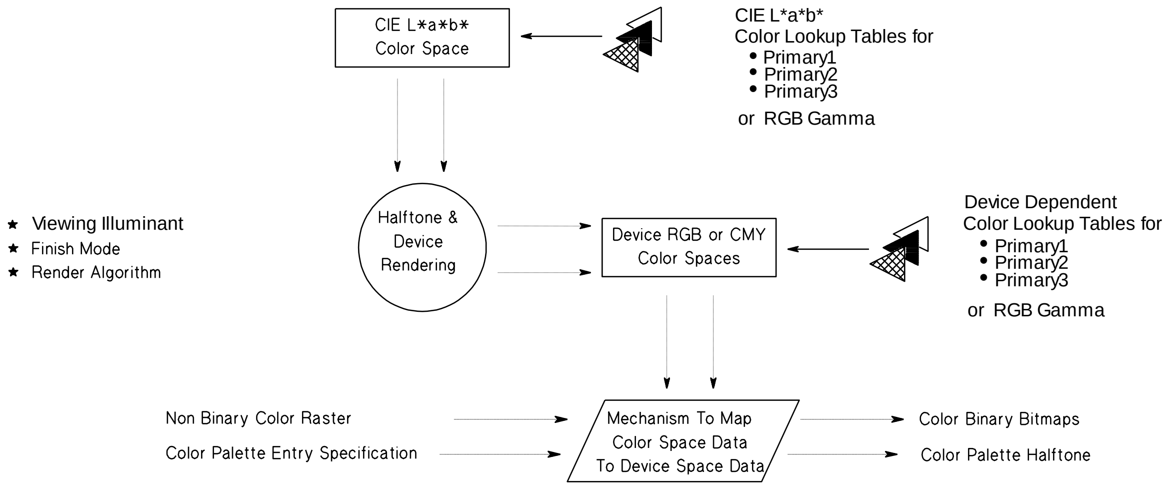
The diagram below shows the two forms of processing for device-dependent color.


Render Algorithm
Halftone & Device Rendering
Device RGB or CMY Color Spaces
Device Dependent Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
Non Binary Color Raster
Color Palette Entry Specification
Mechanism To Map Color Space Data To Device Space Data
Color Binary Bitmaps
Color Palette Halftone

Device-dependent Pipeline

CIE L*a*b* Pipeline

One of the device-independent color spaces supported by the pipeline is CIE L*a*b*. The effect of CIE L*a*b* on the color processing pipeline is shown below:



Device-Independent Pipeline—CIEL*a*b*

The CIE L*a*b* color pipeline builds on the device-dependent pipeline. The following PCL parameters can be used:

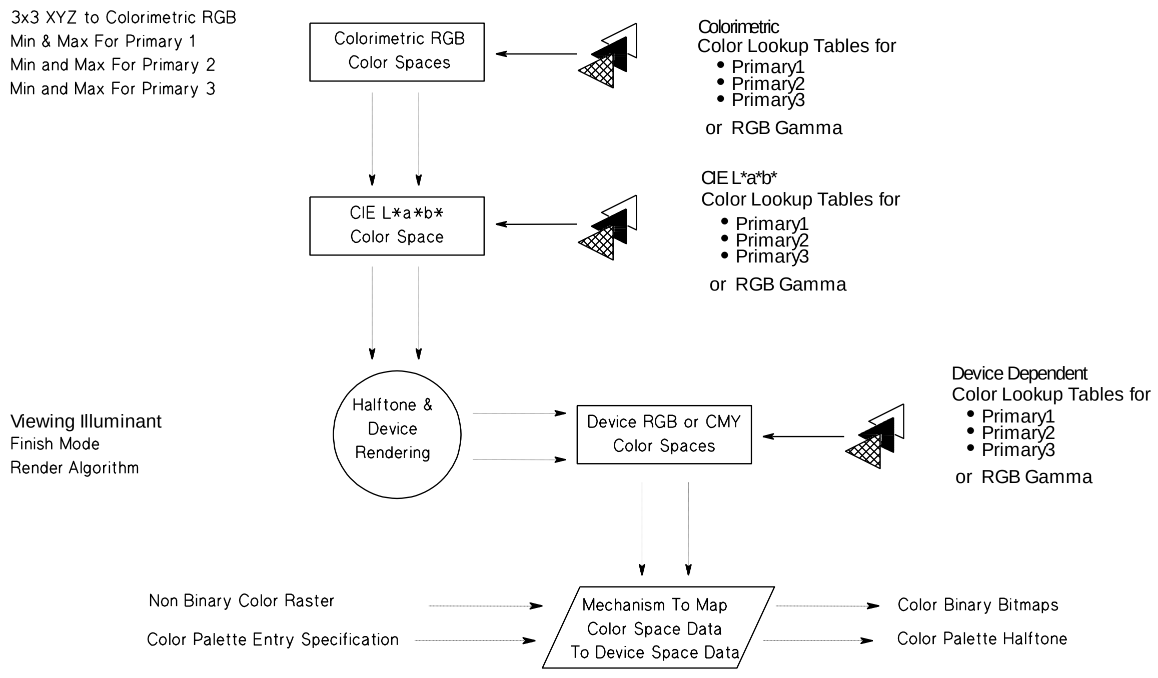
- Rendering algorithm (optional; restricted to the four aforementioned types)
- RGB gamma correction (optional)
- Device-dependent lookup tables for each of the three primaries (optional)
- CIE L*a*b* color lookup tables for each of the three primaries (optional)
- Viewing Illuminant (optional)
- Finish Mode (optional)

Note that different types of color lookup tables allow manipulation of color data in either CIE L*a*b* or device-dependent color spaces. This allows the user to manipulate color data differently in different color spaces. For example, increasing color saturation is easier in CIE L*a*b* than Device RGB because a* and b* can be increased equally. The same increase in R, G and B would increase lightness rather than saturation.

There are PCL language defaults for all the state variables. Default settings given in the discussions of each variable.

Colorimetric RGB Pipeline

The next level in the color processing pipeline includes the colorimetric RGB spaces:


3x3 XYZ to Colorimetric RGB
Min & Max For Primary 1
Min and Max For Primary 2
Min and Max For Primary 3
Colorimetric RGB Color Spaces
Colorimetric Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
CIE L*a*b* Color Space
CIE L*a*b* Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
Viewing Illuminant
Finish Mode
Render Algorithm
Halftone & Device Rendering
Device RGB or CMY Color Spaces
Device Dependent Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
Non Binary Color Raster
Color Palette Entry Specification
Mechanism To Map Color Space Data To Device Space Data
Color Binary Bitmaps
Color Palette Halftone

Device-Independent Pipeline—Colorimetric RGB

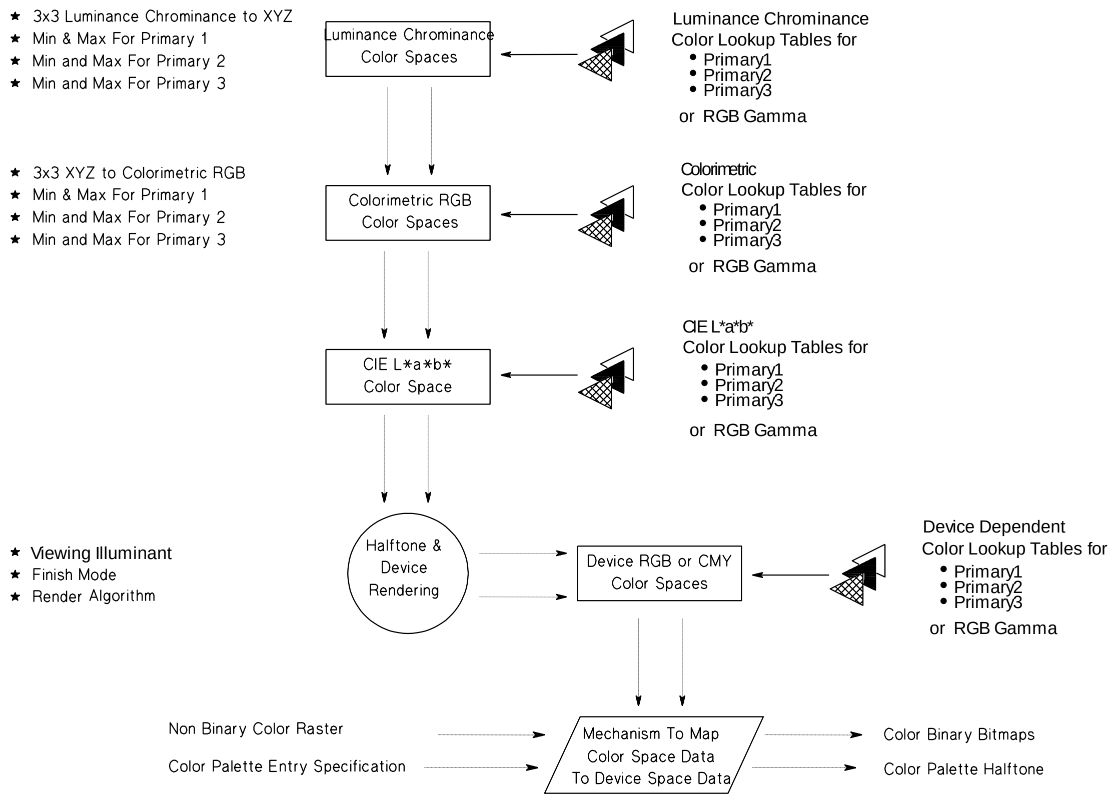
The addition of Colorimetric RGB spaces is merely an extension of the CIE L*a*b* pipeline. Again, this level can have color lookup tables attached to each primary, as well as different color lookup tables attached to different color spaces further down the pipeline. The only new extensions to this level are the following additional parameters provided by the Configure Image Data command (*Esc*v#W*).

- XYZ coordinates for each primary
- xy chromaticity coordinates for the white point
- Gamma and Gain values
- Minimum and Maximum data ranges for each primary

An example of this type of color usage is color raster from a Sony Trinitron or Hitachi Color monitor. In fact, with the above parameters, any color monitor can have its output transformed into device-independent color, and therefore achieve color appearance matching from monitor to output device.

Luminance-Chrominance Pipeline

The Luminance-Chrominance portion of the pipeline is similar in extension and data content to the Colorimetric RGB pipeline entry.


★ 3x3 Luminance Chrominance to XYZ
★ Min & Max For Primary 1
★ Min and Max For Primary 2
★ Min and Max For Primary 3
Luminance Chrominance Color Spaces
Luminance Chrominance Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
★ 3x3 XYZ to Colorimetric RGB
★ Min & Max For Primary 1
★ Min and Max For Primary 2
★ Min and Max For Primary 3
Colorimetric RGB Color Spaces
Colorimetric Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
CIE L*a*b* Color Space
CIE L*a*b* Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
★ Viewing Illuminant
★ Finish Mode
★ Render Algorithm
Halftone & Device Rendering
Device RGB or CMY Color Spaces
Device Dependent Color Lookup Tables for
• Primary1
• Primary2
• Primary3
or RGB Gamma
Non Binary Color Raster
Color Palette Entry Specification
Mechanism To Map Color Space Data To Device Space Data
Color Binary Bitmaps
Color Palette Halftone

Device-Independent Pipeline—Luminance-Chrominance

14.2 Simple Color Mode

The Simple Color command (*Esc*r#U*) allows color selection from a fixed palette. RGB or CMY raster data must be sent by plane (*Esc*b#V*) as well as by row (*Esc*b#W*). The last plane in each row is sent by *Esc*b#W*; all other planes are sent by *Esc*b#V*. In Simple Color mode, the pixel encoding mode is always indexed planar.

Simple Color *Esc * r # u/U*

Creates a fixed-size palette, whose color specification cannot be modified.

Value(#)	=	-4	4 planes, device KCMY palette
	=	-3	3 planes, device CMY palette
	=	1	Single plane black and white palette
	=	3	3 planes, device RGB palette
Default	=	1	
Range	=	-4,-3,1,3	

DEVICE NOTE: DJ500C, PJXL300, DJ1200C, and Color LJ support values of -3,1, 3. DJ560C supports all four values.

DEVICE NOTE: For backward compatibility with PaintJet and PaintJet XL, PaintJet XL300 also supports values of 1, 2, 3, 4 if the raster resolution is specified as 90 or 180 dpi. In this case the palette is programmable. (This should not be documented externally).

The absolute value of the value field indicates the number of planes per row of raster data to be sent. The number of entries in the new palette is 2^n, with index values 0 to $2^n - 1$. For example a 4-plane palette has 16 entries, with index numbers 0 to 15.

This command creates a new palette and overwrites the current palette. PCL and HP-GL/2 commands that modify the palette (*CR*, *NP*, *PC*, *Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, *Esc*t#I*) are locked out. When a Simple Color palette is popped from the stack (*Esc*p#P*), it cannot be modified; and raster pixel encoding mode reverts to indexed planar.

A value field of 1 creates a 2-entry black and white default LaserJet palette.

A value field of 3 creates an 8-entry Device RGB palette (compatible with a PCL Imaging Mode palette, but not an HP-GL/2 default (IN) palette).

A value field of -3 creates an 8-entry palette in Device CMY color space.

A value field of -4 supports 4-plane Device KCMY color. Plane 1 is the black pen, and planes 2, 3, and 4 are CMY planes.

The four Simple Color palettes are shown below.

Single Plane (value = 1)

Index	Color
0	White
1	Black

3-Plane RGB (value = 3)

Index	Color
0	Black
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White

3-Plane CMY (value = -3)

Index	Color
0	White
1	Cyan
2	Magenta
3	Blue
4	Yellow
5	Green
6	Red
7	Black

4**-Plane KCMY (value = -4)**

Index	Black Pen	Color Pen
0	White	White
1	Black	White
2	White	Cyan
3	Black	Cyan
4	White	Magenta
5	Black	Magenta
6	White	Blue
7	Black	Blue
8	White	Yellow
9	Black	Yellow
10	White	Green
11	Black	Green
12	White	Red
13	Black	Red
14	White	Black
15	Black	Black

14.3 PCL Imaging Mode

PCL Imaging Mode, entered by the Configure Image Data (CID) command (*Esc*v#W*), creates a variable-sized, programmable palette. It provides halftoning at the printer, with multiple color spaces, pixel encoding modes, and reprogrammable palettes.

Configure Image Data (CID) *Esc * v # W [binary data]*

The CID command provides configuration information for palette creation and raster data transmission in a single escape sequence by performing the following:

- Designates the color space for the default palette
- Designates the size of the palette to be created
- Provides data for the resolution of color-space-specific values into device-specific values
- Designates the format of raster data
- Designates how primary components are combined to yield the raster representation.

Value(#)	=	Number of data bytes	
Default	=	NA	
Range	=	Short form: 6 bytes	Command is ignored for invalid configurations and
		Long form: >6 bytes	the data discarded; value field signs are ignored.

The data fields must contain byte-aligned binary data, not ASCII.

This command has two forms: the 6-byte ***short form***, and the ***long form*** consisting of these six bytes plus additional information specific to the color space.

Common 6-Byte Header

Both the short and the long forms of the CID command use the common 6-byte header shown below, regardless of which color space is specified. The meaning of the header data fields may vary according to the color space.

Byte	15 (MSB) 8	7 0 (LSB)	Byte
0	Color space (UBYTE)	Pixel encoding mode (UBYTE)	1
2	Bits / index (UBYTE)	Bits / primary #1 (UBYTE)	3
4	Bits / primary #2 (UBYTE)	Bits / primary #3 (UBYTE)	5

Byte 0: Color Space

This byte specifies one of the following color spaces:

Value	Color Space
0	Device RGB (default)
1	Device CMY
2	Colorimetric RGB spaces
3	CIE L*a*b*
4	Luminance-Chrominance spaces

DEVICE NOTE: DJs support only a value of 0.

Colorimetric RGB spaces are based on the 1931 standard 2-degree observer and specified by CIE xy chromaticity coordinates. They use the standard D6500 viewing illuminant and a 45 degree illumination model with a 0 degree collector geometry for reflective data.

*CIE L*a*b** is the CIE 1976 Uniform Color Space based on the 1931 standard 2-degree observer, and using a 45 degree illumination model with a 0-degree collector geometry for reflective data. The viewing illuminant is the standard D6500 illuminant.

Luminance-Chrominance spaces are a 3x3 linear transformation from Colorimetric RGB. Like CIE L*a*b*, achromatic data is in one channel and chromatic data shares the other two channels.

Byte # 1: Pixel Encoding Mode

Designates the format in which raster data is transmitted and interpreted.

Value	Pixel Encoding Mode	Restrictions
0	Indexed by Plane (default)	1, 2, 3, 4, 5, 6, 7, 8 bits per index
1	Indexed by Pixel	1, 2, 4, or 8 bits per index
2	Direct by Plane	1 bit per primary (3-bit color) — RGB or CMY only
3	Direct by Pixel	8 bits per primary (24-bit color) — All color spaces

The possible combinations are shown below:

	Plane	Pixel
Indexed	mode 0	mode 1
Direct	mode 2	mode 3

One plane, or one bit per pixel, is needed for each power of two colors in the palette. For example, a 256-color palette requires eight planes — or eight bits per pixel ($2^8 = 256$).

MODE 0: INDEXED BY PLANE

In mode 0 (default), successive planes of data are sent for each raster row. A plane contains one bit for each pixel in a row. A pixel is not fully defined until it has received all the planes for that row. The planes in a row form index numbers that define a pixel by selecting a palette entry. Assuming 3 bits per index, the underlined column of bits below is the palette index for pixel 3 of row 1 (i1 is LSB; i3 is MSB). Note that Transfer Raster by Plane (*Esc*b#V*) is used for all the planes in a row except the last, which uses Transfer Raster by Row (*Esc*b#W*).

*Esc*b#V*	row 1	plane 1	i1 i1 ***i1*** i1 i1 ...
*Esc*b#V*		plane 2	i2 i2 ***i2*** i2 i2 ...
*Esc*b#W*		plane 3	i3 i3 ***i3*** i3 i3 ...
*Esc*b#V*	row 2	plane 1	i1 i1 i1 i1 i1 ...

EXAMPLE:

*ESC*v6W 00 00 03 08 08 08*	# Binary data is represented in hex. Color space is set to RGB, pixel encoding mode to 0, palette size to 8 (3 planes), last 3 bytes set black and white references.
*ESC*r1A*	# Start raster.
*ESC*b1V10110000...*	# Transfer plane 1 (first bit for each pixel in first row). Combining each bit with corresponding bit in other planes forms the palette index number for that pixel.
*ESC*b1V01110000...*	# Transfer plane 2 (the second bit for each pixel in the row).
*ESC*b1W10101000...*	# Transfer plane 3 (the third bit for each pixel in the row)and move to the next row.

MODE 1: INDEXED BY PIXEL

In mode 1, each pixel in a row is fully specified before any bits are sent for the next pixel. The bits for each pixel form a palette index number. Assuming four bits per index, the underlined block below is the palette index for pixel 2 of row 1 (i1 is LSB).

*Esc*b#W*	row 1	i4 i3 i2 i1 ***i4 i3 i2 i1*** ...
*Esc*b#W*	row 2	i4 i3 i2 i1 i4 i3 i2 i1 ...
*Esc*b#W*	row 3	i4 i3 i2 i1 i4 i3 i2 i1 ...

EXAMPLE:

*ESC*v6W 00 01 04 04 04 04*	# Binary data is represented in hex. Color space set to RGB, pixel encoding mode to 1, palette size to 16 (4 bits to address palette index). Last 3 bytes set black and white references.
*ESC*r1A*	# Start raster.
*ESC*b1W45*	# Most significant nibble selects palette index 4 for the first pixel. Second pixel is set to index 5. Move to the next row.
*ESC*b1W6A*	# First pixel is index 6, second pixel is index 10. Move to the next row
*ESC*b1W03*	# First pixel is index 0, second pixel is index 3. Move to the next row.

MODE 2: DIRECT BY PLANE

In mode 2, the color raster data for each row is downloaded by sequential planes; but the pixel color is directly specified, rather than forming an index into the palette. The underlined block below defines the actual primaries for pixel 4 of row 1.

*Esc*b#V*	row 1	red plane	`r r r` <u>**r**</u> `r r r...`
*Esc*b#V*		green plane	`g g g` <u>**g**</u> `g g g...`
*Esc*b#W*		blue plane	`b b b` <u>**b**</u> `b b b...`
*Esc*b#V*	row 2	red plane	`r r r r r r r...`

EXAMPLE:

*ESC*v6W 00 02 01 01 01 01*	# Binary data is represented in hex. Color space is set to RGB, pixel encoding mode to 2. Palette size is ignored. Last three bytes are always 1 for this mode.
*ESC*r1A*	# Start raster.
*ESC*b1V10110000...*	# Transfer plane for primary 1. Each bit turns on or off the red primary for the pixel defined by the corresponding bits in each plane.
*ESC*b1V01110000...*	# Transfer plane for primary 2. Each bit turns on or off the green primary for the pixel defined by the corresponding bits in each plane.
*ESC*b1W10101000...*	# Transfer plane for primary 3 and move to the next row. Each bit turns on or off the blue primary for the pixel defined by the corresponding bits in each plane.

MODE 3: DIRECT BY PIXEL

In mode 3, the color raster data is downloaded pixel by pixel (as in mode 1), but each pixel directly specifies each color component (as in mode 2). Assuming Device RGB space with 8 bits per primary, the underlined block below defines the actual color primaries for pixel 1 of row 2.

*Esc*b#W*	row 1	r7-r0 g7-g0 b7-b0...
*Esc*b#W*	row 2	<u>***r7-r0 g7-g0 b7-b0***</u>...
*Esc*b#W*	row 3	r7-r0 g7-g0 b7-b0...

EXAMPLE:

*ESC*v6W 00 03 00 08 08 08*	# Binary data is represented in hex. Color space is set to RGB, pixel encoding mode to 3. Palette size is ignored. Send 8 bits to address each primary of a pixel.
*ESC*r1A*	# Start raster.
*ESC*b3W 45 06 30*	# Each byte sets the primary value for the first pixel and moves to the next row.

Byte # 2: Number Of Bits Per Index

In all pixel encoding modes, this byte sets the size of the palette to $2^{\text{number of bits per index}}$.

In pixel encoding modes 0 and 1 (indexed), where raster data is interpreted as indices into a palette, this value specifies the number of bits required to access all palette entries.

In pixel encoding modes 2 and 3 (direct), this value determines palette size, but has no effect on the specification of raster data.

Bytes # 3: Number Of Bits For Primary 1
Bytes # 4: Number Of Bits For Primary 2
Bytes # 5: Number Of Bits For Primary 3

These three bytes affect their respective primaries in the same way. For simplicity, the number of a primary is referred to as "#x" in the description below.

In device-dependent color spaces, these bytes affect the black and white references in pixel encoding modes 2 and 3 (direct). In Device RGB, the black reference for primary #x is 0 and the white reference is $2^{(\text{number of bits for primary \#x})}$ - 1). In Device CMY, these references are reversed.

	Short Form	Long Form
Pixel Encoding Modes 0 and 1	• Sets white reference for primary #x to is $2^{(\text{number of bits for primary \#x})}$ - 1) • Sets the black reference to 0 • A value of 0 defaults the B/W refernces • Values > 15 are clamped to 15	• This byte is ignored
Pixel Encoding Mode 2	• This byte is the number of bits (data planes to be sent) needed to specify primary #x • Sets the white reference for the palette (not the CID default palette) • Sets the black reference to 0	• This byte is the number of bits (data planes to be sent) needed to specify primary #x
Pixel Encoding Mode 3	• This byte is the number of bits needed to specify primary #x • Sets the white reference for the palette (not the CID default palette) • Sets the black reference to 0	• This byte is the number of bits needed to specify primary #x

Short Form of CID

The common 6-byte CID header is called the ***Short Form***. By changing the value of byte 0, the short form can specify the following five color spaces (shown in the order they occur in the color pipeline):

- **Device RGB** *Esc*v6W[0x00,...]*

- **Device CMY** *Esc*v6W[0x01,...]*

- **CIE L*a*b*** *Esc*v6W[0x03,...]*

  CIE L*a*b* allows the following data ranges. Hue is preserved if out-of-range data is clipped.

  L* = 0.0 to 100.0
  a* = -100.0 to 100.0
  b* = -100.0 to 100.0

- **Colorimetric RGB (SMPTE RGB)** *Esc*v6W[0x02,...]*

  Non-linear SMPTE RGB with a 2.2 gamma and 1.0 gain is the default Colorimetric RGB color space. The short form allows the following ranges:

  R = 0.0 to 1.0
  G = 0.0 to 1.0
  B = 0.0 to 1.0

- **Luminance-Chrominance (YUV)** *Esc*v6W[0x04,...]*

  YUV, which is a linear transformation from SMPTE RGB, is the default Luminance-Chrominance color space. The short form allows the following ranges:

  Y = 0.0 to 1.0
  U = -0.89 to 0.89
  V = -0.70 to 0.70

Data Range Scaling

White and black references define the encoding range for device-dependent color spaces. However, device-independent color spaces require input data pre-scaled to the range 0-255. For example, to use the short form for the default YUV color space, the input data must have the following ranges:

Y	=	0.0 to 1.0
U	=	-0.89 to 0.89
V	=	-0.70 to 0.70

The user must linearly scale (y = mx + b) the input data to the range 0-255:

Y	=	0 (0.0) to 255 (1.0)
U	=	0 (-0.89) to 255 (0.89)
V	=	0 (-0.70) to 255 (0.70)

Long Form of CID

There is also a ***Long Form*** of the CID command for each color space designated by byte #0 in the common header. In device-independent color spaces, the long form can specify primaries other than the defaults provided by short form. For example, a Sony Trinitron RGB primary base can be selected for Colorimetric RGB instead of the default non-linear SMPTE RGB primaries.

Device RGB (Long Form)

The long form for Device RGB (value field is 18) provides explicit entry of black and white references (range is -32767 to 32767). Black and white references are used when specifying the primary components of new palette entries (*Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#I*). Black and white references have no effect on CID default palette colors.

NOTE: The short form for Device RGB defaults each primary's black reference to 0 and white reference to 2^n - 1, where *n* is the number of bits for that primary.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color space	Pixel encoding mode	1
2	Bits per index	Bits per primary #1	3
4	Bits per primary #2	Bits per primary #3	5
6	White reference for primary #1 (sint16)		7
8	White reference for primary #2 (sint16)		9
10	White reference for primary #3 (sint16)		11
12	Black reference for primary #1 (sint16)		13
14	Black reference for primary #2 (sint16)		15
16	Black reference for primary #3 (sint16)		17

Device CMY (Long Form)

The long form for Device CMY (value field is 18) provides explicit entry of black and white references (range is -32767 to 32767). Black and white references are used when specifying the primary components of new palette entries (*Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#I*). Black and white references have no effect on the default CID palette colors.

NOTE: The short form for Device CMY defaults each primary's white reference to 0 and black reference to 2^n - 1, where n is the number of bits for that primary.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color space	Pixel encoding mode	1
2	Bits per index	Bits per primary #1	3
4	Bits per primary #2	Bits per primary #3	5
6	White reference for primary #1 (sint16)		7
8	White reference for primary #2 (sint16)		9
10	White reference for primary #3 (sint16)		11
12	Black reference for primary #1 (sint16)		13
14	Black reference for primary #2 (sint16)		15
16	Black reference for primary #3 (sint16)		17

CIE L*a*b* (Long Form)

The long form for CIE L*a*b* allows a different data range than the short form defaults:

L* = 0.0 to 120.0 (over short form by 20.0)
a* = -159.0 to 128.0 (under short form by -59.0 and over short form by 28.0)
b* = -120.0 to 80.0 (under short form by 20.0)

NOTE: Although the data ranges may extend beyond the default data ranges specified in the short form of the CID command for CIE L*a* b*, the printer will clip the data to the short form data ranges

Maximum and minimum values are specified for each primary. Floating point data must be linearly scaled (y = mx + b) to the range 0-255.

Since a* and b* have no theoretical limits, L*a*b* data may be sent outside CID constraints. Then data is clipped to preserve hue and compressed to the device's printable gamut.

The white point is based on the standard D6500 illuminant.

The following single-precision, 32-bit floating point specification, which is used for device-independent color floating point specifications is fully compliant with the *IEEE Floating Point Formats*:

31	30 23	22 0
Sign	Exponent	Fractional Portion

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color space	Pixel encoding mode	1
2	Bits per Index	Bits per primary #1	3
4	Bits per primary #2	Bits per primary #3	5
6	Minimum L* value (most significant word)**		7
8	Minimum L* value (least significant word)**		9
10	Maximum L* value (msw)		11
12	Maximum L* value (lsw)		13
14	Minimum a* value (msw)		15
16	Minimum a* value (lsw)		17
18	Maximum a* value (msw)		19
20	Maximum a* value (lsw)		21
22	Minimum b* value (msw)		23
24	Minimum b* value (lsw)		25
26	Maximum b* value (msw)		27
28	Maximum b* value (lsw)		29

** Floating point format

Colorimetric RGB (Long Form)

The long form for Colorimetric RGB allows specifications other than the default non-linear SMPTE RGB with a 2.2 gamma and 1.0 gain. Each RGB primary and the white point is specified in the CID data field by chromaticity coordinates (CIE xy). The tristimulus luminance Y value of the white point is assumed to be 100% and is therefore not specified. For color spaces that are linear transformations from CIE XYZ tristimulus coordinates, gamma and gain are set to 1.0; otherwise they are set appropriately. Colorimetric RGB spaces can be used for any monitor having primaries specified as CIE xy chromaticity coordinates with white point, such as the Sony Trinitron or Hitachi Color Monitor.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color space	Pixel encoding mode	1
2	Bits per Index	Bits per primary #1	3
4	Bits per primary #2	Bits per primary #3	5
6	x Chromaticity for red primary (most significant word)**		7
8	x Chromaticity for red primary (least significant word)**		9
10	y Chromaticity for red primary (msw)		11
12	y Chromaticity for red primary (lsw)		13
14	x Chromaticity for green primary (msw)		15
16	x Chromaticity for green primary (lsw)		17
18	y Chromaticity for green primary (msw)		19
20	y Chromaticity for green primary (lsw)		21
22	x Chromaticity for blue primary (msw)		23
24	x Chromaticity for blue primary (lsw)		25
26	y Chromaticity for blue primary (msw)		27
28	y Chromaticity for blue primary (lsw)		29
30	x Chromaticity for white point (msw)		31
32	x Chromaticity for white point (lsw)		33
34	y Chromaticity for white point (msw)		35
36	y Chromaticity for white point (lsw)		37
38	Gamma for red primary (msw)		39
40	Gamma for red primary (lsw)		41
42	Gain for red primary (msw)		43
44	Gain for red primary (lsw)		45
46	Gamma for green primary (msw)		47
48	Gamma for green primary (lsw)		49
50	Gain for green primary (msw)		51
52	Gain for green primary (lsw)		53
54	Gamma for blue primary (msw)		54
56	Gamma for blue primary (lsw)		57
58	Gain for blue primary (msw)		59
60	Gain for blue primary (lsw)		61
62	Minimum red value (msw)		63
64	Minimum red value (lsw)		65
66	Maximum red value (msw)		67
68	Maximum red value (lsw)		69
70	Minimum green value (msw)		71
72	Minimum green value (lsw)		73
74	Maximum green value (msw)		75
76	Maximum green value (lsw)		77
78	Minimum blue value (msw)		79
80	Minimum blue value (lsw)		81
82	Maximum blue value (msw)		83
84	Maximum blue value (lsw)		85

** Floating point format

Luminance-Chrominance (Long Form)

The long form for Luminance-Chrominance allows color spaces other than the default YUV, such as Kodak Photo YCC, the proposed JPEG and TIFF6.0 YCrCb standard, YES, and YIQ. These Luminance-Chrominance spaces are linear transforms from a Colorimetric RGB space defined by CIE xy chromaticity coordinates and white point. The tristimulus luminance y value of the white point is assumed to be 100% and is therefore not specified.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color space	Pixel encoding mode	1
2	Bits per index	Bits per Primary #1	3
4	Bits per primary #2	Bits per Primary #3	5
6	Encoding for primary # 1 R (most significant word)**		7
8	Encoding for primary # 1 R (least significant word)**		9
10	Encoding for primary # 1 G (msw)		11
12	Encoding for primary # 1 G (lsw)		13
14	Encoding for primary # 1 B (msw)		15
16	Encoding for primary # 1 B (lsw)		17
18	Encoding for primary # 2 R (msw)		19
20	Encoding for primary # 2 R (lsw)		21
22	Encoding for primary # 2 G (msw)		23
24	Encoding for primary # 2 G (lsw)		25
26	Encoding for primary # 2 B (msw)		27
28	Encoding for primary # 2 B (lsw)		29
30	Encoding for primary # 3 R (msw)		31
32	Encoding for primary # 3 R (lsw)		33
34	Encoding for primary # 3 G (msw)		35
36	Encoding for primary # 3 G (lsw)		37
38	Encoding for primary # 3 B (msw)		39
40	Encoding for primary # 3 B (lsw)		41
42	Minimum primary #1 value (msw)		43
44	Minimum primary #1 value (lsw)		45
46	Maximum primary #1 value (msw)		47
48	Maximum primary #1 value (lsw)		49
50	Minimum primary #2 value (msw)		51
52	Minimum primary #2 value (lsw)		53
54	Maximum primary #2 value (msw)		55
56	Maximum primary #2 value (lsw)		57
58	Minimum primary #3 value (msw)		59
60	Minimum primary #3 value (lsw)		61
62	Maximum primary #3 value (msw)		63
64	Maximum primary #3 value (lsw)		65
66	x Chromaticity for red primary (msw)		67
68	x Chromaticity for red primary (lsw)		69
70	y Chromaticity for red primary (msw)		71
72	y Chromaticity for red primary (lsw)		73
74	x Chromaticity for green primary (msw)		75
76	x Chromaticity for green primary (lsw)		77
78	y Chromaticity for green primary (msw)		79
80	y Chromaticity for green primary (lsw)		81
82	x Chromaticity for blue primary (msw)		83
84	x Chromaticity for blue primary (lsw)		85
86	y Chromaticity for blue primary (msw)		87
88	y Chromaticity for blue primary (lsw)		89
90	x Chromaticity for white point (msw)		91
92	x Chromaticity for white point (lsw)		93
94	y Chromaticity for white point (msw)		95
96	y Chromaticity for white point (lsw)		97

98	Gamma for red primary (msw)	99
100	Gamma for red primary (lsw)	101
102	Gain for red primary (msw)	103
104	Gain for red primary (lsw)	105
106	Gamma for green primary (msw)	107
108	Gamma for green primary (lsw)	109
110	Gain for green primary (msw)	111
112	Gain for green primary (lsw)	113
114	Gamma for blue primary (msw)	115
116	Gamma for blue primary (lsw)	117
118	Gain for blue primary (msw)	119
120	Gain for blue primary (lsw)	121

** Floating point format

Examples of CID Usage

The following examples illustrate CID long and short forms for each color space. For clarity, data is shown as ASCII rather than binary, and the CID command (*Esc*v#W*) is shown as "CID". The following format is used:

CID (data , data , ...)

Device RGB (dRGB) or Device CMY (dCMY)

Short Form

CID(0,1,8,8,8,8)	dRGB, 8 bits/pixel indexed
CID(1,1,8,8,8,8)	dCMY, 8 bits/pixel indexed

EXAMPLE: The short form of the CID command, as a C function, can look like this:

```
short_cid(Color_mode, Pixel_mode, BitsperIndex, BitsperColor_1, BitsperColor_2,
    BitsperColor_3)
{
    int Color_mode, Pixel_mode, BitsperIndex, BitsperColor_1, BitsperColor_2, BitsperColor_3;

    printf("\033*v6W%c%c%c%c%c%c",Color_mode, Pixel_mode, BitsperIndex,
    BitsperColor_1, BitsperColor_2,                 BitsperColor_3);

}
```

Long Form

```
CID( 0,1,8,8,8,8,          dRGB, 8 bits/pixel indexed
     100,100,100           white reference
     0,0,0)                black reference


CID( 1,1,8,8,8,8,          dCMY, 8 bits/pixel indexed
     0,0,0                 white reference
     100,100,100)          black reference
```

CIE L*a*b*

Short Form

```
CID( 3,3,0,8,8,8 )         Lab, direct 8 bits/primary
```

Long Form

```
CID( 3,3,0,8,8,8,          Lab, direct 8 bits/primary
     0.0, 100.0,           L* data encoding
     -100.0, 100.0,        a* data encoding
     -100.0, 100.0)        b* data encoding
```

Non-Linear SMPTE RGB, 2.2 Gamma, 1.0 Gain

Short Form

```
CID( 2,3,0,8,8,8 )         RGB, direct 8 bits/primary
```

Long Form

```
CID( 2,3,0,8,8,8,          RGB, direct 8 bits/primary
     0.64, 0.34,           |
     0.31, 0.60,           | chromaticity coordinates
     0.16, 0.07,           | for RGB & White Point
     xxxx, yyyy,           |
     2.2, 1.0,             *
     2.2, 1.0,             * gamma and gain for RGB
     2.2, 1.0,             *
     0.0, 1.0,             @
     0.0, 1.0,             @ data range encoding
     0.0, 1.0 )            @
```

Non-Linear Sony Trinitron

Short Form

<< not applicable >>

Long Form

```
CID( 2,3,0,8,8,8,            RGB, direct 8 bits/primary
     0.62, 0.34,             |
     0.30, 0.58,             | chromaticity coordinates
     0.15, 0.09,             | for RGB & White Point
     xxxx, yyyy,             |
     2.3, 1.19,              *
     2.3, 1.19,              * gamma and gain for RGB
     2.3, 1.19,              *
     0.0, 255.0,             @
     0.0, 255.0,             @ data range encoding
     0.0, 255.0 )            @
```

YUV Chrominance-Luminance Space

Short Form

```
CID( 4,3,0,8,8,8 )           YUV, direct 8 bits/primary
```

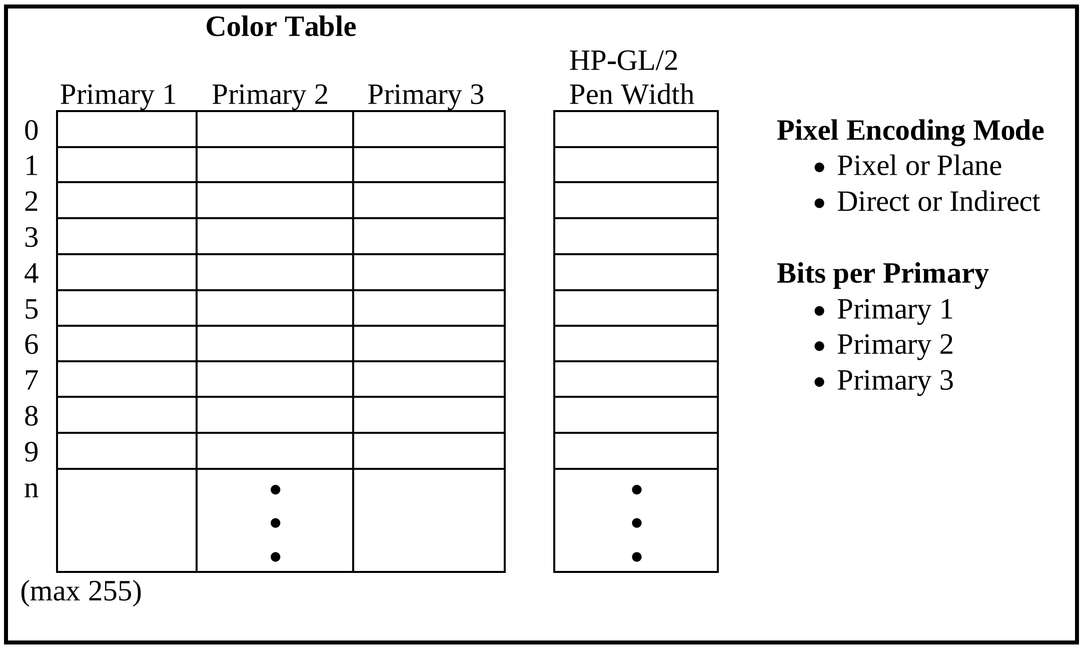
YUV Chrominance-Luminance with Sony Trinitron

Long Form

```
CID( 2,3,0,8,8,8             YUV, direct 8 bits/primary
     0.30,0.59,0.11,
     -0.30,0.59,0.89,        # 3x3 YUV matrix
     0.70,-0.59,-0.11,       #
     0.0,255.0,              +
     -227.0,227.0,           + data encoding
     -179.0,179.0,           +
     0.62,0.34,              |
     0.30,0.58,              | chromaticity
     0.15,0.09,              | coordinates
     xxxx,yyyy,              |
     2.3,1.19,               *
     2.3,1.19,               * gamma and
     2.3,1.19)               * gain for RGB
```

14.4 Palette Operations

A palette is a collection of colors that are selected by their index numbers. The figure below illustrates a palette. Each palette entry associates an index number with three primary color components. For HP-GL/2 purposes only, a pen width is also associated with each palette entry.



In non-raster mode, the current palette contains all the colors available to the printer. In raster mode, indexed color selection uses the palette, but direct selection does not.

Only one palette at a time can be active. A new palette is created by the following conditions:

- Power-up and *EscE* (default black and white palette)
- Simple Color (*Esc*r#U*)
- Configure Image Data (*Esc*v#W*)
- HP-GL/2 (*IN, BP*)

Default palettes are created by all the PCL color modes (Black and White, Simple Color, PCL Imaging, and HP-GL/2 Imaging). The active palette may be modified in the PCL and HP-GL/2 Imaging modes, but not in the Simple Color or Black and White modes. The active palette is automatically transferred when switching between PCL and HP-GL/2 contexts.

Although only one palette can be active at any one time, multiple inactive palettes can exist on the system by means of the following two mechanisms:

- Palette ID
- Palette stack

Simple Color Palettes

The Simple Color command (*Esc*r#U*) provides a quick way to select colors from a fixed, non-reprogrammable palette.

The Simple Color command (*Esc*r#U*) command overwrites the current palette with one of the fixed palettes below. PCL and HP-GL/2 commands that modify a palette entry (*NP*, *CR*, *PC*, *Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, *Esc*t#I*) are locked out. A popped simple color palette cannot be modified; pixel encoding mode reverts to "index by plane". To create a modifiable palette, *IN*, *BP*, *Esc*v#W* (CID), or *Esc*d#W* (Palette Configuration) must be sent.

As shown below, a Simple Color value field of 1 creates a black and white palette. A value of 3 creates an 8-pen palette in Device RGB color space. A value of -3 creates an 8-pen palette in Device CMY color space. A value field of -4 supports 4-plane Device KCMY color; plane 1 is the black pen, and planes 2, 3, and 4 are CMY planes. All of these palettes are fixed and nonprogrammable.

Black and White (value = 1)

Index	Color
0	White
1	Black

RGB Palette (value = 3)

Index	Color
0	Black
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White

CMY Palette (value = -3)

Index	Color
0	White
1	Cyan
2	Magenta
3	Blue
4	Yellow
5	Green
6	Red
7	Black

4-**Plane KCMY (value = -4)**

Index	Black Pen	Color Pen
0	White	White
1	Black	White
2	White	Cyan
3	Black	Cyan
4	White	Magenta
5	Black	Magenta
6	White	Blue
7	Black	Blue
8	White	Yellow
9	Black	Yellow
10	White	Green
11	Black	Green
12	White	Red
13	Black	Red
14	White	Black
15	Black	Black

CID Color Palettes

The Configure Image Data command (*Esc*v#W*) creates one of the three programmable palettes below. Palette entries may be reprogrammed with different colors by PCL (*Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*) or HP-GL/2 (*CR*, *NP*, *PC*) commands.

Device RGB Palettes

The CID-specified black and white references have no effect on the default palettes below. However, if a CID palette entry is reprogrammed with a different color, the white and black references are used to specify the primary components of the new color.

Bits/Index = 1

Index	Color
0	White
1	Black

Bits/Index = 2

Index	Color
0	Black
1	Red
2	Green
3	White

Bits/Index = 3 through 8

Index	Color
0	Black
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White
n > 7	Black

Device CMY and Device-Independent Palettes

A CID command specifying either Device CMY or a device-independent color space will create the same default palettes. This is because device-independent colors are resolved into the printer's native space, Device CMY.

Bits/Index = 1

Index	Color
0	White
1	Black

Bits/Index = 2

Index	Color
0	White
1	Cyan
2	Magenta
3	Black

Bits/Index = 3 through 8

Index	Color
0	White
1	Cyan
2	Magenta
3	Blue
4	Yellow
5	Green
6	Red
7	Black
n > 7	Black

HP-GL/2 Palettes

As shown below, default HP-GL/2 palettes are different than default PCL palettes. The following table shows the default palettes established in HP-GL/2. Like a default CID palette, a default HP-GL/2 palette can be modified in either PCL or HP-GL/2 contexts (*Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I* or *NP*, *PC*, *CR*). *IN* and *BP* always establish the 8-pen palette; *NP* may be used to enlarge it.

Two Pens

Pen Number	Color
0	White
1	Black

Four Pens

Pen Number	Color
0	White
1	Black
2	Red
3	Green

Eight Pens

Pen Number	Color
0	White
1	Black
2	Red
3	Green
4	Yellow
5	Blue
6	Magenta
7	Cyan
n >7	Black

Default HP-GL/2 Palettes

Palette Configuration

Palettes created in HP-GL/2 or by the Configuration Image Data command (*Esc*v#W*) are programmable — their size and entries can be modified by the PCL and HP-GL/2 palette programming commands (*Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#I, CR, NP, PC*). Devices that do not support HP-GL/2 or the CID command can use the Palette Configuration command (*Esc*d#W*) to create a palette, which then can be modified by the PCL palette programming commands (*Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#I*).

Palette Configuration   *Esc * d # w/W*

Configures the device to receive a variable-sized, programmable color palette; sets red, green, and blue component ranges; and optionally specifies the components of each palette entry.

Value(#)	=	Number of data bytes
Default	=	NA
Range	=	9 to 2^{32}-1

DEVICE NOTE: DJ660C and 850C support only a range of 9 to 32767.

Data fields must contain byte-aligned binary data, not ASCII. Invalid configurations are ignored and the data discarded. Signs are ignored, and extra bytes are discarded. The command is ignored for value fields less than 9 and the specified number of data bytes discarded.

This command:

- Configures a programmable palette.
- Specifies the size of the palette (byte 2).
- Specifies maximum and minimum ranges for red, green, blue components (bytes 3-8).
- Optionally specifies palette entries by defining their color components (bytes 9...).

Byte 2 indirectly sets the programmable palette size by specifying the last index entry. Since index numbers start with 0, palette size is (last_index + 1).

Bytes 3-8 specify the maximum and minimum ranges for the red, green, and blue components in device-dependent RGB color space. The minimum is the 0% point of a color component; the maximum is the 100% point. This range scales color component values; but default palette entries are not scaled. The following formula performs the scaling:

scaled_value=(value-minimum)*(255/(maximum-minimum)).

Some or all of the entries may be specified using three bytes for each entry: one each for the red, green, and blue components. Unspecified palette entries are defaulted and not affected by the minimum and maximum range values. The palette programming commands (*Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*) can subsequently modify any palette entry. Default entries are:

Index	Color
0	White
1	Black
2	Red
3	Green
4	Yellow
5	Blue
6	Magenta
7	Cyan
>7	Black

Excess data (e.g., the number of palette entries exceeds the specified number of entries) is ignored and the data bytes discarded. A palette entry that is started but not completely specified (e.g., only the red component is specified) is ignored and the default entry retained; the data bytes for that entry are discarded.

Reset (*EscE*), Simple Color (*Esc*r#U*), and Configure Image Data (*Esc*v#W*) override this command's configuration. Configure Raster Data (*Esc*g#W*) has no effect on the palette.

The format of the binary data for this command is as follows:

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Color space (UBYTE)	Format # (UBYTE)	1
2	Last palette entry index (UBYTE)	Min range value for red component (UBYTE)	3
4	Max range value for red component (UBYTE)	Min range value for green component (UBYTE)	5
6	Max range value for green component (UBYTE)	Min range value for blue component (UBYTE)	7
8	Max range value for blue component (UBYTE)	Red component of palette entry *n* (UBYTE)	3n+6
3n+7	Green component of palette entry *n* (UBYTE)	Blue component of palette entry *n* (UBYTE)	3n+8

Byte 0: Color Space

Value = 1 Device Dependent RGB

A value of 1 configures for device-dependent RGB palette data. The command is ignored for other values and the specified number of data bytes discarded.

Byte 1: Format

Value = 1 Unsigned 8-bit Integer Data

A value of 1 specifies unsigned 8-bit integer values for the device-dependent RGB palette data. The command is ignored for other values and the specified number of data bytes discarded.

Byte 2: Last Palette Entry Index

Value = 1 to 255 (command is ignored for unsupported values)

Specifies the index of the last palette entry. This also determines the number of entries: index numbers start with 0, so palette size is (last_index + 1). If palette size would exceed the device limit, the device stores up to its limit and ignores all entries beyond.

DEVICE NOTE: DJ850C supports only 255.

Byte 3: Minimum Range Value for Red Component

Value = 0 to 254 (command is ignored for unsupported values)

Specifies the minimum red component value. Values less than the minimum are clamped to the minimum. Red values less than or equal to the minimum represent 0% red in RGB color space.

DEVICE NOTE: DJ850C supports only 0.

Byte 4: Maximum Range Value for Red Component

Value = 1 to 255 (command is ignored for unsupported values)

Specifies the maximum red component value. Values greater than the maximum are clamped to the maximum. Values greater than or equal to the maximum represent 100% red in RGB color space. The command is ignored and the specified number of data bytes discarded if the maximum red value is not greater than the minimum.

DEVICE NOTE: DJ850C supports only 255.

Byte 5: Minimum Range Value for Green Component

Value = 0 to 254 (command is ignored for unsupported values)

Specifies the minimum green component value. Values less than the minimum are clamped to the minimum. Values less than or equal to the minimum represent 0% green in RGB space.

DEVICE NOTE: DJ850C supports only 0.

Byte #6: Maximum Range Value for Green Component

Value = 1 to 255 (command is ignored for unsupported values)

Specifies the maximum green component value. Values greater than the maximum are clamped to the maximum. Values greater than or equal to the maximum represent 100% green in RGB color space. The command is ignored and the specified number of data bytes discarded if the maximum green value is not greater than the minimum.

DEVICE NOTE: DJ850C supports only 255.

Byte #7: Minimum Range Value for Blue Component

Value = 0 to 254 (command is ignored for unsupported values)

Specifies the minimum blue component value. Values less than the minimum are clamped to the minimum. Values less than or equal to the minimum represent 0% blue in RGB color space.

DEVICE NOTE: DJ850C supports only 0.

Byte #8: Maximum Range Value for Blue Component

Value = 1 to 255 (command is ignored for unsupported values)

Specifies the maximum blue component value. Values greater than the maximum are clamped to the maximum. Values greater than or equal to the maximum represent 100% blue in RGB color space. The command is ignored and the specified number of data bytes discarded if the maximum blue value is not greater than the minimum.

DEVICE NOTE: DJ850C supports only 255.

Bytes 9, 12, etc: Red Component of Palette Entry *n*

Value = 0 to 255 (Command is ignored for unsupported values)

Specifies the red component of the nth programmable palette entry. Unless all three RGB components are specified, the default entry retained. The red component is scaled by the maximum and minimum values specified by bytes 3-4 using the following formula:

scaled_red_value=(red_value-red_minimum)*(255/(red_maximum-red_minimum)).

Bytes 10, 13, etc: Green Component of Palette Entry *n*

Value = 0 to 255 (Command is ignored for unsupported values)

Specifies the green component of the *n*th programmable palette entry. Unless all three RGB components are specified, the default entry is retained. The green component is scaled by the maximum and minimum values specified by bytes 5-6 using the following formula:

scaled_green_value = (green_value - green_minimum)*(255/(green_maximum -green_minimum)).

Bytes #11, 14, etc: Blue Component of Palette Entry *n*

Value = 0 to 255 (Command is ignored for unsupported values)

Specifies the blue component of the *n*th programmable palette entry. Unless all three RGB components are specified, the default entry is retained. The blue component is scaled by the maximum and minimum values specified by bytes 7-8 using the following formula:

scaled_blue_value = (blue_value - blue_minimum)*(255/(blue_maximum - blue_minimum)).

Programming the Palette

Other than the default black and white palette or the Simple Color Palette (*Esc*r#U*), palette entries can be modified in HP-GL/2 (*CR*, *NP*, *PC*), or by the PCL commands below. In the discussion below, "components" refer to the color space primaries. For example, if the current color space is CIE L*a*b*, component one is L*, component two is a*, and component three is b*.

Color Component One *Esc * v # A*

Specifies the first component of any new palette color entry.

Value(#)	=	First Component
Default	=	0
Range	=	-32767 to 32767 (fractional values allowed; up to 4 decimal places)

This command affects the palette entry designated by Assign Color Index (*Esc*v#I*), then reset to 0.

Color Component Two *Esc * v # B*

Specifies the second component of any new palette color entry.

Value(#)	=	Second Component
Default	=	0
Range	=	-32767 to 32767 (fractional values allowed; up to 4 decimal places)

This command affects the palette entry designated by Assign Color Index (*Esc*v#I*), then reset to 0.

Color Component Three *Esc * v # C*

Specifies the third component of any new palette color entry.

Value(#)	=	Third Component
Default	=	0
Range	=	-32767 to 32767 (fractional values allowed; up to 4 decimal places)

This command affects the palette entry designated by Assign Color Index (*Esc*v#I*), then reset to 0.

Assign Color Index *Esc * v # I*

Assigns the three current color components to the specified palette index number.

Value(#)	=	Index Number
Default	=	0
Range	=	0 to $2^{\text{(current palette size)}}$ - 1

This command assigns the color components specified by the above commands to the designated index. After assignment, the three color components are reset to 0. Values greater than the palette size make no assignment, but reset the three components to 0.

Saving Palettes

Creating a palette overwrites the active palette. The default palettes created by a power-on, reset, the Simple Color command, the CID command, or the *IN* command overwrite the active palette. A copy of the active palette may be saved by "pushing" it onto the palette stack, and restored by "popping" it from the stack. The active palette is unaffected when its copy is pushed onto the stack.

Push / Pop Palette *Esc * p # P*

Pushes or pops the palette from the palette stack. The last item pushed is the first item popped.

Value(#)	=	0	Push (save) palette
	=	1	Pop (restore) palette
Default	=	0	
Range	=	0,1	

A value of 0 pushes only a copy of the active palette, which is itself unaffected. A value of 1 pops the most recently pushed palette, which then overwrites the active palette and becomes the new active palette. Any palette with same ID number as the popped palette is destroyed.

Pushing a palette saves the following parameters:

- Color components for each palette entry
- Pen widths (for HP-GL/2 use)
- Color space specification
- Black and white references
- Number of bits per index
- Pixel encoding mode
- Number of bits per primary
- Gamma correction
- Viewing illuminant
- Color lookup tables
- Render algorithm
- Downloaded dither matrix
- Monochrome print mode

Pushing a palette does not save the following parameters. A popped palette, which will use the current values of these variables, may therefore be different with respect to these variables from the original pushed palette.

- Foreground color
- Color components: 1st, 2nd, and 3rd

Stack depth is limited by memory. Attempts to push with insufficient memory cause an out-of-memory error. Attempts to pop from an empty stack are ignored.

DEVICE NOTE: PJ XL300 limits stack size to 256.

Macros can push and pop palettes. A palette that was popped in an executed macro remains in effect at the end of the macro; but this is not true for called or overlaid macros.

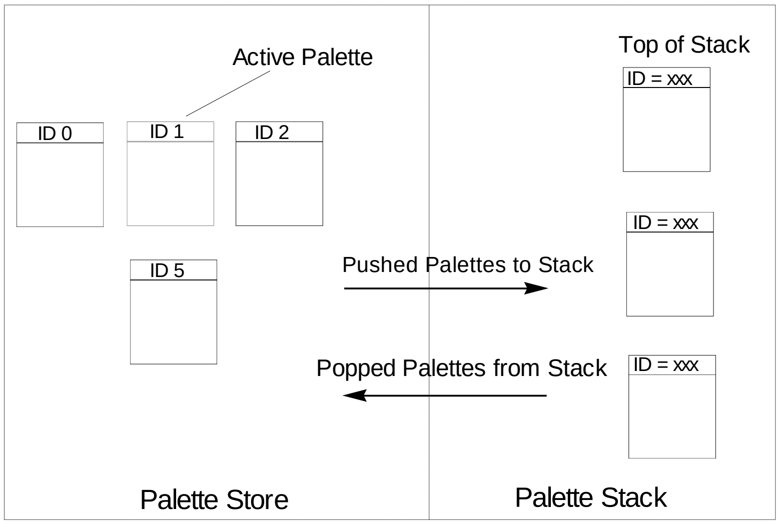
A reset or PJL entry empties the stack and overwrites the current palette with a non-programmable black and white palette. The *IN* and *DF* commands overwrite the current palette with the default HP-GL/2 palette, but have no effect on the palette stack.

The Palette Control ID (*Esc&p#I*) and Select Palette (*Esc&p#S*) commands do not affect palettes on the stack. The Palette Control command (*Esc&p#C*) can delete all palettes on the stack.

Managing Palettes by ID

All palettes have a unique ID (identification number). The default black and white palette created on power-up or *EscE* has an ID of 0.

Palette management by ID lets applications have multiple palettes. As shown below, multiple palettes can exist in two areas: the palette *stack* and the palette *store*. The stack holds palettes that are pushed via a Push/Pop Palette command; the store holds palettes having palette IDs.



Palettes on the stack may not be selected by ID, since only a copy of a palette is pushed onto the stack; the original palette and ID remain in the palette store. A palette popped from the stack goes into the palette store, becomes the new active palette and assumes the ID of the previous active palette, which is overwritten. Only one palette at a time may be active.

Management by ID allows applications to tag data, have multiple raster configurations, and have palettes for different color spaces — all without reconfiguring the active palette. For example, one palette can be created for PCL text, one for HP-GL/2 primitives, one for simple raster, and one for 24-bit raster. The application can then switch between palettes according to what is being sent to the printer.

Selecting a new active palette changes the PCL graphics state. As described previously in detail, besides color entries, a palette also contains the graphics state at the time the bitmap representation of the palette colors was created. This guarantees color reproduction integrity by insuring that the same color specification triplet always produces the same bitmap representation.

As described below, the Select Palette (*Esc&p#S*), Palette Control (*Esc&p#C*), and Palette Control ID (*Esc&p#I*) commands implement the three basic operations of management by ID:

- Selection of the active palette.
- Deletion of palettes.
- Copying of palettes.

Select Palette *Esc & p # s/S*

Selects a new active palette by ID. The previous active palette is unchanged.

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

This command activates the designated palette in the palette store. The command is ignored if no palette with that ID exists. The designated ID is saved as the *palette select ID* in the current modified print environment.

This command can be used to de-select the active palette and select as the new active palette a palette created by the Palette Control command (*Esc&p#C*). For example, to copy the active palette to an ID of 44 and select the new palette to use or modify, send *Esc&p44i6c44S*

When a palette creation command is received, such as Configure Image Data (*Esc*v#W*), Simple Color (*Esc*r#U*), or an HP-GL/2 *IN*, the created palette overwrites the active palette and is assigned the current *palette select ID*, which is unchanged.

A popped palette overwrites the active palette and is assigned the current *palette select ID*.

EscE resets the *palette select ID* value to 0 and deletes all palettes in the palette stack and palette store, including the active palette, which is replaced by a default PCL fixed black and white palette with a *palette select ID* value of 0.

Macros affect the *palette select ID*. A called or overlaid macro saves the *palette select ID* as well as a copy of the active palette; upon macro exit, the restored palette becomes active palette with the saved ID (an existing palette with this ID is deleted). An executed macro does not save the ID or the active palette: changes remain in effect.

Palette Control ID *Esc & p # i/I*

Specifies the ID to be used by the Palette Control command.

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

The ID specified by this command is saved as the *palette control ID* value in the current modified print environment and is used by the Palette Control command (*Esc&p#C*).

EscE or power-up reset the *palette control ID* to 0, which is then the default black and white palette ID.

Macros affect the *palette control ID*. A called macro saves the value and then restores it upon exit. An executed macro does not save the value; changes remain in effect at exit. An macro overlay copies the value before resetting to 0, and restores it at exit.

Palette Control *Esc & p # c/C*

Provides a mechanism for copying and deleting palettes.

Value(#)	=	0	Delete all palettes except those in the stack (active palette deleted).
	=	1	Delete all palettes in the stack (active palette is not affected).
	=	2	Delete palette (specified by Palette Control ID).
	=	6	Copy the active palette to ID specified by Palette Control ID.
Default	=	0	
Range	=	0,1,2,6	

A value of 0 deletes all palettes except those on the palette stack. The active palette is replaced by the default black and white palette (ID = 0). The *palette control ID* is not used.

A value of 1 clears the palette stack. The active palette is unaffected, and the *palette control ID* is not used.

A value of 2 deletes the palette with the specified *palette control ID* if it exists; otherwise the command is ignored. For example, to delete palette 53, send *Esc&p53i2C*. If the active palette's ID is specified, the active palette is replaced by the default black and white palette. This option does not change the *palette control ID* value.

NOTE: When the active palette is replaced by the default black and white palette, the graphics state associated with the previous palette is also replaced.

A value of 6 creates a copy of the active palette. The copy receives the ID specified by the last Palette Control ID command. For example, to copy the active palette to a palette with an ID of 14, send *Esc&p14i6C*. The copied palette overwrites any palette that already has an ID equal to the *palette control ID*. The copied palette does not become the active palette. The command is ignored if a palette is to be copied to its own ID.

The Palette Control command provides a way of managing system memory by deleting palettes in either the stack or store that are no longer in use.

14.5 Foreground Color

All PCL marking entities utilize "foreground" color, which is selected by *Esc*v#S* from the current palette. Raster color interacts with foreground color.

Foreground Color *Esc * v # s/S*

Sets the foreground color to the specified index of the current palette.

Value(#)	=	Index Number Into Current Palette
Default	=	Black
Range	=	0 to $2^{(\text{current palette size})}$ - 1)

Out-of-range values are mapped into a new index via modulo (palette size). For example, a foreground color index of 10 with a current palette size of 8 is mapped to 2 (i.e., 10 modulo 8). If the current palette was created under HP-GL/2, the index is mapped according to the HP-GL/2 mapping function.

Foreground color affects the following PCL page marking primitives.

- Text characters change to foreground color (including underlining)
- Solid or monochrome patterned rectangular area fills (rules)
- Monochrome patterns (except HP-GL/2)
- Raster images

The following are not affected:

- User-Defined color patterns (format 1 download patterns)
- HP-GL/2 marking primitives (HP-GL/2 uses "selected pen", ignores foreground color).

NOTE: Foreground color interacts with color raster images. In the printer, all color raster is resolved into three binary raster planes of CMY. Foreground color is applied to these planes, modifying the color image. For no interaction, set foreground color to black when sending color raster images.

After a foreground color is selected, changing any of the following WILL NOT change foreground color until a new Foreground Color command (*Esc*v#S*) is issued.

- Active Palette
- Configure Image Data command
- Render Algorithm
- User Defined Dither Matrix
- Gamma Correction
- Color Lookup Tables
- Viewing Illuminant

DEVICE NOTE: DJ1600C includes Monochrome Print Mode in the above list.

Monochrome Print Mode (*Esc&b#M*) immediately maps foreground color to its equivalent gray, and deselection of Monochrome Print Mode immediately returns foreground color to its color equivalent. *

* DEVICE NOTE: As noted above, this is not true for the DJ1600C.

DEVICE NOTE: On DJ5xx's and 8xx's, foreground color does not affect raster color; On DJ560, foreground color is used only for text.

14.6 Halftone Algorithms

Color printers may use halftone algorithms to modify a printed image by changing the way pixels are rendered. For example, a dither pattern can convert from 256 colors to 8 colors, or a color image can be converted to monochrome. The Render Algorithm command (*Esc*t#J*) provides a choice of existing algorithms or a user-defined pattern created with the Download Dither Matrix command (*Esc*m#W*).

A single page may have multiple render algorithm calls during page composition, but only one is in effect at any given time.

Render Algorithm *Esc * t # j/J*

Selects the algorithm to be used for rendering page marking entities on a given page.

Value(#)	=	0	Continuous tone (otherwise, device best)
	=	1	Snap to primaries
	=	2	Snap black to white and other colors to black
	=	3	Device-best dither
	=	4	Error diffusion
	=	5	Device-best dither (monochrome)
	=	6	Error diffusion (monochrome)
	=	7	Cluster ordered dither
	=	8	Cluster ordered dither (monochrome)
	=	9	User-defined dither
	=	10	User-defined dither (monochrome)
	=	11	Ordered dither
	=	12	Ordered dither (monochrome)
	=	13	Noise ordered dither
	=	14	Noise ordered dither (monochrome)
Default	=	3	
Range	=	0 to 14 (values 1,2,9,10 are ignored for device-independent color)	

DEVICE NOTE: Since noise dither is its device best, Color LJ does not support values 13 and 14.

DEVICE NOTE: PJXL300 allows only single N×N matrices (N=2,4,8,16).

DEVICE NOTE: On Color LJ, in device-independent color spaces algorithms 1,2,9, and 10 are not accessible. The following substitutions are made: 1,2,9 ⇒ 3 and 10 ⇒ 5.

DEVICE NOTE: On Color LJ, in non-raster mode algorithms 4 and 6 are not accessible. The following substitutions are made: 4 ⇒ 3 and 6 ⇒ 5.

Device Best: This is the render method that HP believes will provide the best output for a particular device in most cases. Note, however, that the recommended dither pattern varies with the image, the intended use of the image, and the subjective judgment of the user.

Continuous Tone: Tone variations are rendered as a continuous series of tones without using a dither pattern.

Snapping to Primaries: Converts each component of a color specification to its corresponding primary color. For example: assuming 8 bits per primary, an RGB input value greater than or equal to 128 snap to 255; a value less than 128 snaps to 0.

Snapping Black to White: Converts black to white and all other colors to black. Input primaries equal to a black specification are converted to a white specification, and other color specifications for the input primaries are converted to the black specification.

Ordered (Clustered) Dither: A pixel is intensified at a point (x,y) depending on the desired intensity, I(x,y), and on an $n \times n$ dither matrix, D, where

$$i = x \text{ modulo } n$$
$$j = y \text{ modulo } n$$

For RGB color spaces, if $I(x,y) < D(i,j)$, the point corresponding to (x,y) is intensified; otherwise it is not. The intensity of each primary is determined according to this scheme.

Error Diffusion: The input primaries of a given (x,y) pixel are printed at the closest density available and the local error is propagated to the unprinted neighboring pixels. The error is multiplied by empirically-determined coefficients before it is propagated. Error diffusion applies only to raster data printed using the Configure Image Data command (*Esc*v#W*).

NOTE: Error diffusion applies only to raster data with Configure Image Data (*Esc*v#W*).

Monochrome Rendering: Generates a gray value from the three primaries that is computed according to the NTSC standard, which for Device RGB is: $\text{gray} = 0.3 \times \text{red} + 0.59 \times \text{green} + 0.11 \times \text{blue}$.

User Defined Dither: The input primaries are compared against differently dimensioned dithers (e.g. M x N), which may be different for each primary.

NOTE: Since it is impossible to characterize a printer for all possible dither algorithms, color correction is invalid whenever a user-defined or monochrome user-defined halftone or render algorithm (*Esc*t9J* or *Esc*t10J*) is selected.

14.7 User-Defined Dithers

User-defined dither matrices can optimize the printer's output. A user-defined matrix is defined in additive colors (RGB values). The dither matrix pixels are defined in terms of device-dependent resolution. The Download Dither Matrix command can create a dither matrix for one or all three primary colors, in effect providing halftone screens. This command allows several options:

- A matrix can be defined for each color plane, or the same matrix for all three planes.
- The height and width of the dither cell can be set. When using separate matrices for each plane, the size of the dither cells for each plane can be different (e.g., a 4 x 4 pixel cell for red, a 4 x 6 cell for green, and a 6 x 8 cell for blue).
- The data bytes for each pixel of the cell are downloaded. Each data byte determines a threshold: every pixel with a value greater than or equal to the threshold is turned on and every pixel with a value less than the threshold is not turned on.

If the render algorithm is user-defined dither (*Esc*t9J* or *Esc*t10J*) when a device-independent color space is active, the default algorithm is used until an explicit algorithm selection of a non-user-defined dither matrix, or until a device-dependent Configure Image Data command (*Esc*r#W*) is received.

User-defined dithers are not rotated according to logical page orientation; dithers are rendered in a device-dependent fashion according to print/scan direction.

Download Dither Matrix   *Esc * m # W [Data]*

Specifies a single user-defined dither matrix for all three primaries, or three user-defined matrices (one for each primary), each of which may have different sizes and contents.

Value(#)	=	Number of bytes of byte-aligned binary data in the data field
Default	=	0
Range	=	7 to 2^{32} -1 (command is ignored for values of 0 to 6; values larger than 2^{32} -1 or the device limits are clamped; signs are ignored)

DEVICE NOTE: DJ1200C does not support multi-plane dither matrices.

The downloaded dither matrix can be specified as one matrix which is applied to all three color primaries during rendering of the color, or a downloaded dither matrix can be specified for each primary. When specified for each primary, the matrices may have different sizes and contents. Multiple matrix definitions are specified consecutively.

A downloaded dither matrix is saved and not used until the printer receives a user-defined Render Algorithm (*Esc*t9J* or *Esc*t10J*) command. If the render algorithm is 9 or 10 and the matrix has not been downloaded, the default render algorithm (3) is used.

Since user-defined dithers cannot be used when a device-independent color space is active, specification of a user-defined Render Algorithm (*Esc*t#J*) defaults the render algorithm until the render algorithm is explicitly changed (to other than user-defined) or until a device-dependent color space is specified.

NOTE: The dither matrix must be defined for processing with additive colors (RGB).

The table below shows the format for a dither matrix that is applied to all three color primaries ("uint16" means 16-bit unsigned integer; "ubyte" means unsigned byte).

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Format = 0	Number of planes = 1	1
2	Dither matrix height in pixels (uint16)		3
4	Dither matrix width in pixels (uint16)		5
6	byte #0 (ubyte)	byte #1 (ubyte)	7
8	byte #2 (ubyte)	byte #3 (ubyte)	9
	*		
	*		

Format

This byte should be set to 0.

Number of Planes

This byte designates how many dither matrices are specified by the command. A value of 1 indicates that one matrix is applied to all primaries. A value of 3 indicates that each primary has a separate matrix. The command is ignored and the data discarded for values other than 1 or 3.

Height and Width

These bytes designate the size of the dither matrix in pixels. For example, a height of four and and width of six produces a 4 x 6 dither cell. Values must be non-zero and sized so the matrix contains no more than 32767 bytes; otherwise, the command is ignored and the data discarded. The minimum dither matrix size is 1 x 1.

Data Bytes

After specifying the height and width of the cell, data bytes are sent row-by-row (row-major order. Each data byte contains the normalized probabilities, ranging from 0 to 255, of one cell. For example, a 2 x 2 cell might have no pixels turned on for values of 0 through 50, one pixel on for values of 51 through 102, two pixels on from 103 through 154, three pixels on for 155 through 206, and all four pixels on between 207 and 255.

Each dither matrix must be completely specified. Otherwise, the width and height values may be misinterpreted if multiple matrices are sent.

If the width, height, and data specifications result in an odd number of data bytes, the next matrix specification will begin on an odd byte boundary. No padding is provided for even-byte-aligning.

Multiple Dither Matrices

The *number of planes* field must be 3 to send separate matrices for each primary. Each dither matrix must have its own width and height data fields. As shown below, each matrix specification follows the previous matrix specification.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Format = 0	Number of planes = 3	1
2	Dither matrix height in pixels (uint16)		3
4	Dither matrix width in pixels (uint16)		5
6	byte #0 (ubyte)	byte #1 (ubyte)	7
8	byte #2 (ubyte)	byte #3 (ubyte)	9
	*		
	*		
n	Dither matrix height in pixels (uint16)		n+1
n+2	Dither matrix width in pixels (uint16)		n+3
n+4	byte #0 (ubyte)	byte #1 (ubyte)	n+5
n+6	byte #2 (ubyte)	byte #3 (ubyte)	n+7
	*		
	*		
m	Dither matrix height in pixels (uint16)		m+1
m+2	Dither matrix width in pixels (uint16)		m+3
m+4	byte #0 (ubyte)	byte #1 (ubyte)	m+5
m+6	byte #2 (ubyte)	byte #3 (ubyte)	m+7

Example

This example produces a 4 x 4 dither matrix that is applied to all three color primaries (the number of planes is set to 1). The command is sent as ***Esc*m22W010404B0B1B2B3B4...B15*** (where the first 6 binary bytes are shown as ASCII here for clarity, and B1...B15 indicate the binary byte data). The byte-aligned binary data field (shown as ASCII for clarity) is shown below:

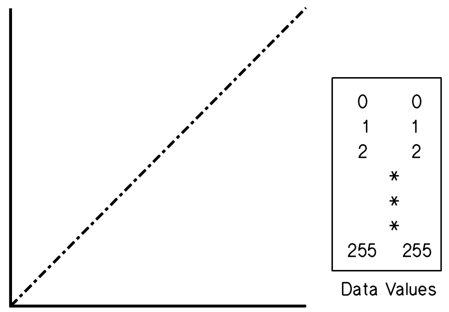
Byte	15(msb) 8	7 (lsb) 0	Byte
0	0	1	1
2	0	4	3
4	0	4	5
6	B0	B1	7
8	B2	B3	9
	*		
	*		
20	B14	B15	21

14.8 Color Lookup Tables

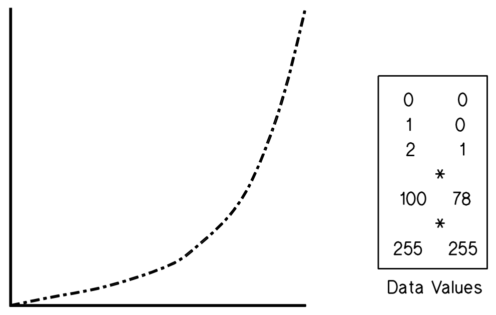
Color lookup tables, which map input data into a new output range based on point-by-point conversions, can modify input data for both device-dependent and device-independent color spaces. A lookup table is specified for each primary. Uses include:

- Highlight and shadow modification
- Saturation and desaturation
- Unique gamma correction curves
- Special effects for tonal correction
- Neutral balancing

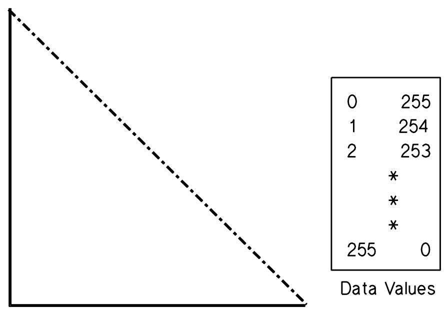
Like the CID command, the first byte of the data field identifies the color space to which the lookup tables will be applied. These tables specify on a point-per-point basis a transformation from an input space of 0...255 into an output space of 0...255. The following diagram illustrates the concept:



Unity Color Lookup Table



Gamma Color Lookup Table



Inversion Color Lookup Table

The unity lookup table is the default for all color spaces in the color processing pipeline; it performs a 1:1 mapping of input to output (e.g., 129 is mapped to 129). The inversion lookup table performs a simple color inversion; for example, it inverts the red primary of a device-dependent RGB color space to create cyan output (255 red ⇒ 0 red, which is 255 cyan).

Color Lookup Tables   *Esc * l # W [binary data]*

Enables and specifies color lookup tables.

Value(#)	=	Number of bytes of binary data
Default	=	0
Range	=	0 or 770 (Command is ignored for other values; signs are ignored)

A value of 0 resets the color lookup tables for each primary to the unity curve (1:1).

This command enables the color lookup tables until an *Esc E*, CID, or another Color Lookup Tables command with a 0 value field is received. RGB gamma correction (*Esc*t#I*) and color lookup tables for device-dependent color spaces are mutually exclusive and overwrite each other.

As shown below, the 256 point-by-point transformation curve for each primary is defined sequentially for a total of 768 bytes, with the additional two bytes for color space and reserved data field.

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	Color Space	Reserved Data Field	1
2	Color Component 1, Index 0	Color Component 1, Index 1	3
4	Color Component 1, Index 2	Color Component 1, Index 3	5
256	Color Component 1, Index 254	Color Component 1, Index 255	257
258	Color Component 2, Index 0	Color Component 2, Index 1	259
260	Color Component 2, Index 2	Color Component 2, Index 3	261
512	Color Component 2, Index 254	Color Component 2, Index 255	513
514	Color Component 3, Index 0	Color Component 3, Index 1	515
516	Color Component 3, Index 2	Color Component 3, Index 3	517
768	Color Component 3, Index 254	Color Component 3, Index 255	769

Byte # 0: Color Space

Value	Color Space
0	Device RGB
1	Device CMY
2	Colorimetric RGB Spaces
3	CIE L*a*b*
4	Luminance-Chrominance Spaces

A color lookup table can be attached to one or more of the color spaces anytime after a CID command. For example, a Luminance-Chrominance space can have four lookup tables specified, namely:

- Device Dependent Space
- CIE L*a*b* space
- Colorimetric RGB space
- Luminance-Chrominance space

EscE or a CID command sets each of the four levels of color lookup tables for each primary to the unity curve.

14.9 Gamma Correction

Color monitors, which are by nature non-linear, appear incorrect when given a linear ramp of some color. Gamma correction improves perceptual correctness by adjusting the brightness or darkness of the color data sent from the monitor to any other non-linear device.

Gamma Correction *Esc * t # i/I*

Specifies the gamma correction to be applied equally for each primary.

Value(#)	=	Gamma number
Default	=	0 (gamma correction off)
Range	=	0.0 to 32767.0 (out-of-range values are clamped to the supported limit)

Assuming 8 bits per primary (256 intensity levels per primary), the corrected intensity for each color primary is calculated as follows:

$$\text{intensity} = ((\text{input value} / 255))^{1/\text{gamma}}) * 255$$

Gamma correction is in terms of device-dependent RGB. This command destroys the contents of device-dependent color lookup tables; and vice versa.

For driver writers, color lookup tables applied separately to each color plane provide better control over the exact gamma function.

NOTE: The default value (0) gives the same result as a gamma value of 1.0, which results in a unity gamma curve.

14.10 Viewing Illuminant

Printed colors undergo a hue shift under different illuminations (e.g., fluorescent, tungsten, or daylight). Colors with spectral characteristics outside the range of an illumination source are not visible under that source; this changes the appearance of mixed colors. The Viewing Illuminant command (*Esc*i#W*) specifies the x and y chromaticities of a relative white point that can be used for a given illuminant. This command can compensate for light sources different from daylight (such as tungsten or fluorescent lighting) if the spectral characteristics of the light are known.

Viewing Illuminant *Esc * i # W [binary data]*

Specifies the relative white point used in the determination of a viewing illuminant condition.

Value(#)	=	Number of binary bytes of data
Default	=	8
Range	=	8 (signs in the value field are ignored)

The binary data field is formatted as follows:

Byte	15 (msb) 8	7 (lsb) 0	Byte
0	x chromaticity white point (lsw)		1
2	x chromaticity white point (msw)		3
4	y chromaticity white point (lsw)		5
6	y chromaticity white point (msw)		7

The above format adheres to the IEEE floating point format as follows:

31	30 23	22 0
Sign	Exponent	Fraction

The PCL default viewing illuminant is D65 (6500K). Below is a table of viewing illuminants and their chromaticity values.

Illuminant		x chromaticity	y chromaticity
Daylight (D65)	(6500K)	0.3127	0.3290
Tungsten	(3200K)	0.4476	0.4074
Cool White Fluorescent	(5630K)	0.3904	0.3914

This command affects only device-independent color. The command acts like a state variable: it is ignored for White/Black, Device RGB, or Device CMY palettes; but it becomes active when a new CID command specifies a device-independent color space.

NOTE: Viewing Illuminant is not available in the CIE L*a*b* color space.

14.11 Page Media Color Commands

Printed output can be modified to look best when printed on different media. There are two page media color commands:

- Monochrome Print Mode
- Finish Mode

Monochrome Print Mode converts each color value to its gray-scale equivalent. This improves throughput, costs less to print, and eliminates waste by providing a draft mode.

Finish Mode allows the user to specify if the output page has either a glossy or non-glossy finish. Transparencies are all printed with a glossy finish.

Monochrome Print Mode *Esc & b # m/M*

Designates either the current rendering mode or a fast gray scale equivalent.

Value(#)	=	0	Print in mixed render algorithm mode
	=	1	Print everything in gray equivalent
Default	=	0	
Range	=	0, 1	

This command must be sent prior to printable data; otherwise, changing the monochrome print mode will close and print the current page. The command may be sent on a page-by-page basis.

Pages printed using the gray-scale equivalent do not use any color and therefore print faster and more economically.

DEVICE NOTE: On DJ1600C, Monochrome Print Mode will not change foreground color to monochrome until a new Foreground Color command is issued.

Finish Mode *Esc & b # f/F*

Determines the finish applied to printed pages.

Value(#)	=	0	matte finish
	=	1	glossy finish
Default	=	0	
Range	=	0, 1	

This command must be sent prior to printable data; otherwise, changing the finish mode will close and print the current page. The command may be sent on a page-by-page basis.