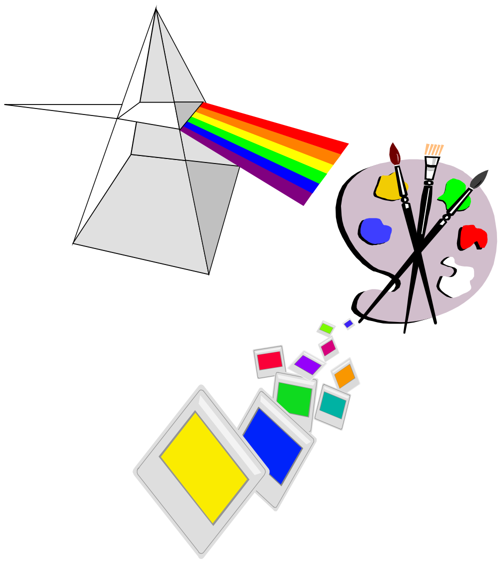
Hewlett-Packard

PCL 5c Color

Programming Quick Reference

for the

HP Color LaserJet 4500, 4550, 8500, and 8550 printers



Developer Technology Services

Third Party Developer Program

Notice

Printing History

This manual was created using Microsoft Word 97 for Windows software.

Printed in the U.S.A.

Version 2.0 November 2000

Trademarks

Table of Contents

About This Guide

This guide provides a quick reference view of programming with PCL 5c in color on HP Color LaserJet 4500, 4550, 8500, and 8550 printers. It starts out by providing a brief overview of color as it relates to the HP Color LaserJet printers. From there, it briefly goes over the key PCL 5c commands that relate to programming color in PCL 5c for the HP Color LaserJet 4500, 4550, 8500, and 8550 printers. Then all of this information is pulled together in brief programming examples that relate the commands to actual working programs.

For more in-depth information, see the PCL Technical Reference manuals or reference the HP LaserJet Developer's Web Site for additional technical information. This site is located at **http://www.hp.com/go/solutions**. Documentation is listed under "LaserJet -- Color.

Note

"Esc" in this document refers to the ASCII escape character which identifies the subsequent string of characters as a print command.

Related Documentation

- HP PCL 5 Printer Language Technical Reference Manual
- HP PCL 5 Comparison Guide
- HP PCL 5 Color Technical Reference Guide
- HP Printer Job Language Technical Reference Manual
- HP Color LaserJet 8550 Developer's Quick Reference Guide
- PCLSample.zip -- zip file containing copies of the PCL sample programs.

Note

Examples were developed using the JetAsm98.exe and srtool.exe which is available on the HP LaserJet Developer's Web Site at **http://www.hp.com/go/solutions** under "LaserJet -- Monochrome."

Color Overview

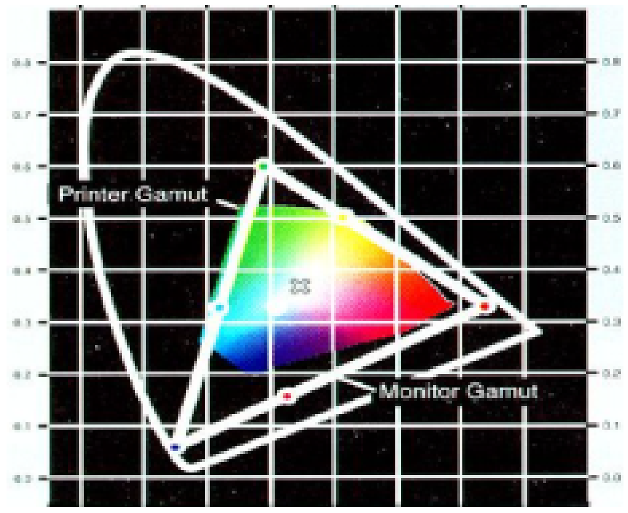
Color Models

Color models (or color spaces) are different ways to describe how color is created or perceived. These may be three or four dimensional whose dimensions or components represent intensity values. There are two major color models that are important to digital imaging: RGB and CMYK. RGB and CMYK are complementary color spaces. RGB (Red, Green, Blue) is a three-dimensional model that corresponds to the way the human eye sees color, that is, with red, green, and blue "cones." It is the color space that describes the color capabilities of photo film and display monitors. RGB is also described as additive and transmitted. Additive refers to the fact that the three primary colors in RGB at 100% (255, 255, 255) combine to create white, the sum of all colors. The transmitted quality of RGB is best exemplified by a color monitor, which creates color when white light strikes the red, green, and blue phosphors on the screen. RGB is capable of creating millions of colors. RGB values are described using numbers where (0,0,0) is black. When any two of the RGB colors combine, they create one of the secondary colors that make up the CMK color model (Cyan, Magenta, and Yellow). The secondary colors produced by the RGB primaries are also called subtractive primaries when used in reference to the CMY color space.

CMY is associated with printed inks or toners on a page. CMY values are described as percentages where (100, 100, 100) represents black. This model is described as subtractive (or absorptive) and reflective. It works the opposite of RGB. When a light source strikes an area of printed magenta toner (ink), all colors are absorbed except the magenta wavelength, which is reflected to the viewer's eye. The color of any part of an image, therefore, results from the frequency of light it reflects and the light-absorbing properties of the toner (or inks) and media. In a subtractive model, a white surface can be thought of as reflecting all wavelengths of visible light. Likewise, a black surface (100% of CMY) absorbs all of them, whereas a green surface (combination of Cyan and Yellow) absorbs (subtracts) all but the green wavelengths.

The combination of CMY at 100% (100,100,100) in theory creates black, but in practice it creates a muddy brown due to limitations of toner (and ink) secondaries. To resolve this, a pure blacK fourth toner is added in order to produce black on a printed page. This results in sharper text and lines using black only, better overall contrast within photographs, better detail and depth in shadow tones, and the ability to replace three color toners with black when printing black only text and images. The letter "K" is used for black to avoid confusion with Blue and because the black component is always considered the "Key" to a set of color separations.

Due to the colorants commonly used today, CMYK (a four-dimensional model) has a more limited gamut (or range of colors) compared to RGB, with a range of only tens of thousands of colors. In fact, the gamut, or range of colors available on a particular device, decreases as you move from the colors in real life to analog film to a digital scanner to a printer. The following graphic compares the color gamut across devices. The outermost area within the boundaries of the horseshoe shape represents all the colors perceivable by the human visual system. The triangle represents the range of colors available on a color monitor and the innermost color area represents the gamut of a color printer. Values on this chart represent the x and y chromaticities (a measure of the combination of both hue and saturation in color produced by lights).



Factors that limit gamut include the following:

- Subtractive versus additive qualities of CMY versus RGB
- Opaqueness of toner (or ink)
- Number of process colors
- Hue, lightness, and chroma (relative colorfulness) in the pigments of the primaries.

Problems with gamma may show up as dark, washed-out, bleached-out, or poor contrasting areas within images.

Black Generation

When converting an RGB or CMY image to CMYK color mode, black generation refers to the values that are generated for the black toner which replaces certain amounts of the cyan, magenta, and yellow toners. A night scene in a city will require more black to keep the shadows dark and crisp. A bright and colorful image will need little or no black to avoid neutralizing the colors. Gray components may use CMY, K only, or CMYK to create the gray images. In a simple example, where cyan is the complement of red, magenta the complement of green, and yellow the complement of blue, there is one and only one unique set of CMY values for any given set of RGB values. Therefore, converting a color from RGB to the CMY model is fairly straight forward. However, things are more complicated when black and real world colorants are considered. A specific shade of purple, for example, may contain anywhere from 10% to 40% black, with the other process colors adjusted accordingly. Thus, for a given set of RGB values, there can be an infinite number of possible CMYK values, resulting in a very complex calculation. In addition, the actual cyan, magenta, and yellow toners are not exact complements of a typical monitor red, green, and blue. These differences must also be compensated for. To manage this, algorithms are developed for the HP Color LaserJet printers to convert from RGB or CMY to CMYK. In addition, the emulation of different CMYK print ink standards are provided to get more accurate color in specific environments. This results in improved colors for specific applications and media.

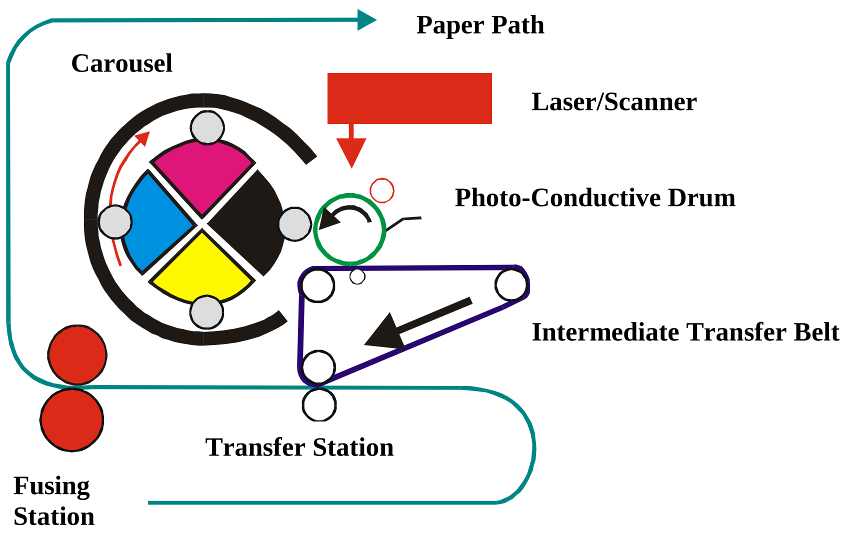
Note

The Pantone Matching System is fundamentally different from the CMYK color model. Pantone colors are recipes for mixing standard printing inks (or toners) of various colors. CMYK colors are proportional, screened percentages of coverage using only CMYK primary colors.

Four Color Printing Process

In the four color printing process, all color images consist of a Cyan layer, Magenta layer, Yellow layer and a Black layer (see the graphic below). When these four layers are precisely aligned on top of one another the final color image is generated. Both the HP Color LaserJet 4500/4550 and 8500/8550 printers use the four color printing process.

In this process, the printer's formatter divides the color image into four color planes. The first color plane (black plane) is generated by the formatter and then written on the photo-conductive drum by the laser/scanner. The black image is then developed on the photo-conductive drum as it passes under the black developer roller in the black toner cartridge. From here, the image is moved onto the intermediate transfer belt. This process is repeated again for the magenta color plane. But, in order to accomplish this, the carousel assembly that holds the four toner cartridges must rotate one-quarter turn to position the magenta toner cartridge's developer roller next to the photo-conductive drum. While the carousal is rotating, the intermediate transfer belt rotates one entire rotation so that the top of the next color plane can be precisely aligned to the top of the color plane previously transferred to the belt. This process is repeated for the cyan and yellow color planes. When all four color planes have been written, developed, and aligned precisely on top of one another on the intermediate transfer belt, the final image is then transferred to paper at the transfer station. From here, the paper is moved to the fusing assembly were the image is permanently bonded to paper and then the paper is placed into the printer's stacker ready for your use.



As you can tell in this printing scenario, it takes time to perform the mechanical activities of rotating the next toner cartridge into position and aligning the top of the color planes on to one another on a single intermediate transfer belt. You can also see why the HP Color LaserJet 4550 printer can print black (monochrome, one color plane images) at 16 pages per minute and color pages (four color plane images) at four pages per minute.

Color Management and sRGB

Color management refers to software that translates the colors of an original image into the truest representation obtainable on the output device. Color management often works from a profile of the output device, typically a digital printer or offset press, and works backward to the source of the material such as a scanner.

For color to be reproduced in a predictable manner across different devices and materials, it has to be described in a way that is independent of the specific mechanisms and materials used to produce it. For instance, color displays and color printers use very different mechanisms for producing color. Traditionally, operating systems have supported color by declaring support for a particular color space (RGB in most cases). Unfortunately, since the interpretation of RGB values varied between devices, color was not reliably reproduced across different devices. To resolve this issue and meet the needs of the very high-end publishing sector that could not be met by the traditional means of color support, the various computer operating systems added support for using International Color Consortium (ICC) profiles to characterize device-dependent colors in a device-independent way. By using the profiles of the input device that created an image, and the output device that displayed or printed the image, a transform could be created that moved the image from the color space of the input device to that of the output device. This resulted in very accurate color and access to the entire color gamut of both devices. However, it also involved the overhead of transporting the profile of the input device with the image and running the image through the transform. In addition, ICC profiles required additional software processes and the need to consciously configure the profiles as well as increased file sizes due to ICC profiles being embedded in image files. This also resulted in gamut mapping issues between devices and incompatibilities between profiles from different companies and between older and newer profiles.

In addition, there are a broad range of users that do not require the level of control and complexity of an embedded ICC color profile mechanism. Also, most existing file formats do not support color profile embedding, and may never do so. There is also a broad range of uses that actually discourages people from appending any extra data to their files. Instead, a single, standard default color space for exchange and interpretation of color data is required. The sRGB color space addresses these issues.

Note

> HP's ICC profiles are available through normal HP software distribution channels. For those who want the additional control available through building their own ICC profiles, there are several vendors of profiling tools available. On the HP Color LaserJet 4500 and 4550 printer, to provide access to the printer's pure primaries and entire available printer gamut, the Vivid mode may be used when profiling the printer, and subsequently when using the ICC workflow. The device CMYK mode can be used on the HP Color LaserJet 8500 and 8550 printers.

The sRGB color space maintains the advantage of a clear relationship with ICC color management systems while minimizing software processes and support requirements. Since the image is in a known color space and the profile for that color space is included within the operating system and display application, this enables end-users to enjoy the benefits of color management without the overhead of larger files. Application developers and users who do not want the overhead of embedding profiles in documents or images can convert them to sRGB. While it may be that profiles buy slightly higher color accuracy, the benefits of using a standard color space far outweigh the drawbacks for a wide range of users. The migration of devices to support the standard color space (sRGB) natively further enhances the speed and quality of the user experience.

The international standard color space sRGB (IEC 61966-2-1) was originally developed and proposed by HP and Microsoft. It is designed to complement current color management strategies by enabling a simple, robust method of handling color in the operating systems, device drivers and the Internet. This solution provides good quality and backward compatibility with minimum transmission and system overhead. Based on a calibrated colorimetric RGB color space well suited to cathode ray tube (CRT) displays, flat panel displays, television, scanners, digital cameras, and printing systems, the sRGB color space can be supported with minimum cost to software and hardware vendors. The four major technical components of the sRGB color space are the standard CRT primaries (HDTV P22 phosphors), the simple gamma value of 2.2, a D65 white point, and its well-defined viewing conditions. The sRGB color space is the default standard for Microsoft Operating Systems, the Worldwide Web, HP color printers, and over 80% of office software. In most cases, the advantages of sRGB are realized without any action by the user. Often users do not even know that the systems and devices are communicating with a calibrated sRGB color space.

Color Calibration

The purpose of color calibration is to maintain the color accuracy of the printer under a wide range of operating conditions, including environmental and consumable usage. It operates by taking multiple measurements of the four color primaries (cyan, magenta, yellow, and black) on the imaging drum and adjusting the output density based on those measurements. Included in those adjustments are any density settings made from the front panel under the calibration menu.

The adjustments themselves are made in two steps. First, the maximum output density for each of the primaries is set to a fixed target value. This sets the amount of toner used for a solid fills and provides the foundation for the second step, halftone correction. The purpose of halftone correction is to guarantee that the entire tone scale (from solid primary to white) behaves as intended. Both the high and low resolution halftones used for the detail and smooth settings are measured and corrected individually.

Color calibration happens automatically and is initiated when any one of the following conditions have been met.

- Power on.
- Warm up from sleep mode if the fuser has cooled significantly. It generally takes 4 hours of sleep before this happens.
- Every 100 color pages printed, but is postponed until the current job has finished printing. Monochrome pages are counted as 1/4th of a color page.
- Replacement of the Imaging Drum.
- Opening the top toner cartridge door, whether or not a cartridge has been replaced.

Color References

There is a wide range of books on color on the market today. The following three offer a breadth of information without requiring a strong background in color to understand the information.

Understanding Desktop Color, by Michael Kieran (ISBN 1-56609-164-0) -- 578 pages

Easy reading with a good overview of basic concepts and terminology of color, design, prepress, and printing.

Measuring Colour, by Dr. R.W.G.Hunt -- (ISBN 0863433871) -- 344 pages

Provides a broad, more in-depth understanding of color measurement principles.

The Reproduction of Colour, by Dr. R.W.G.Hunt -- (ISBN 0863433812) -- 814 pages

Provides a broad, more in-depth understanding of color reproduction in printing, photography, and television.

PCL 5c Support

This section briefly goes over the key PCL 5c commands that relate to programming color in PCL 5c for the HP Color LaserJet 4500, 4550, 8500, and 8550 printers. See the PCL 5 Technical Reference documentation for more information on PCL 5 commands. This is available on the HP Developer Web Site (http://www.hp.com/go/solutions) and is listed under languages for LaserJet – Color.

Developer Guidelines

Job Control

- Support the PJL (Printer Job Language) for language switching. Begin and end all jobs with the Universal Exit Language PJL command *(Esc%-12345*X) and @PJL ENTER LANGUAGE = PCL. The UEL command resets the printer by executing EscE. The ENTER command selects PCL as the language for the current job.
- Never terminate a job in the middle of data transfer, such as *Esc*b#W[data*].

Media

- Support all HP Color LaserJet 4500, 4550, 8500, and 8550 printer media types and sizes.

High-level Objects

- Utilize PCL5's ability to handle patterns, rectangular area fills, and vector graphics.
- Use Foreground Color *(Esc*v#*S) to colorize text, rules, patterns, and black and white raster objects. For best results with raster objects, set foreground color to black.
- Use logical operations *(Esc*l#*O) for MS Windows imaging compatibility.

Fonts

- Select the appropriate symbol set. Support PC-8 and Roman-8 character sets in the US; and PC-850, ECMA 94 and PC-8 Danish/Norwegian in Europe (see the *PCL 5 Comparison Guide*).
- Support the HP Color LaserJet Intellifont and TrueType scalable fonts and downloadable symbol sets.
- Support Autofont/TFM (Tagged Font Metrics).

Programmable Color

- Use 8-bit programmable color *(Esc*v#*W) for text, raster, and HP-GL/2 objects. 8-bit indexed color tends to compress better than 3-bit indexed color, and it allows the printer to do the rendering and scaling.
- Use the printer's internal rendering *(Esc*t#*J) to optimize color quality and utilize different dithering algorithms, scaling, and color controls. Some rendering algorithms are not supported on all printers. See the *PCL 5 Printer Language Technical Reference Manual* and *PCL 5 Comparison Guide* for the list of supported algorithms for each printer.

Optimizing

- Use raster compression methods 1-3 *(Esc*b#*M) and the PE command in HP-GL/2 for smaller output files and faster printing. Compression methods will significantly improve throughput.
- Use high-level HP-GL/2 objects, such as polygons, with a defined fill type. This will produce smaller output files than rasterization. Use the PE command for data compression.
- Use macros to store frequently used templates. This will reduce memory usage (See the *PCL 5 Printer Language Technical Reference Manua*l).
- Always send the Source Raster Width command *(Esc*r#*S) and Raster Height command *(Esc*r#T*) before sending raster to avoid storing unnecessary data and improve printer memory efficiency.
- When scaling an image, use the printer's internal scaling for upscaling or downscaling or send raster at 300 dpi. This will significantly reduce the data sent (see the *PCL 5 Printer Language Technical Reference Manual* for information on HP-GL/2 scaling).
- Rather than using raster for gradient fills, create a series of rectangles that gradually increment the colors. The file size will be substantially reduced.
- Use the printer's internal rendering *(Esc*t#*J) to quickly optimize color quality and utilize different dithering algorithms.
- Use a formfeed instead of *EscE* between pages of a multi-page job. This is more efficient than resetting after every page.
- Save color palettes with Palette IDs (*Esc&p#i6C*) versus pushing and popping palettes (*Esc*p#P*). This provides better control over palettes, reduces the I/O stream, and eliminates the need to reconfigure palettes that are destroyed through push commands.
- For better control over palettes, avoid creating, selecting, pushing, popping, or modifying palettes within macros.

PCL 5c Premises

- A palette always exists
- A palette is shared between PCL and HP-GL/2
- Foreground color is selected from the palette that is currently active when the Foreground color command is sent.
- Color commands are applied to the currently active palette
- Multiple palettes may exist, but only one can be active at any given time
- The current pen is selected from the currently active palette
- Foreground color, current pen, and pattern always exists
- Source and Pattern transparency selections are applied against complete colors, not dithered colors

Color Spaces

The HP Color LaserJet 4500, 4550, 8500, and 8550 printer supports the following color spaces:

- Device Dependent Black (K) -- gray scale or monochrome.
- Device Dependent RGB (Red, Green, Blue).
- Device Dependent CMY (Cyan, Yellow, Magenta).
- Standard RGB (sRGB).

Note

> The HP Color LaserJet 4500, 4550, 8500, and 8550 Color Processing Architecture requires support for the standard RGB (sRGB) color space in place of cRGB. Therefore support is dropped for cRGB and use of the CID command to set the color space to cRGB will set the color space to sRGB.

The following table provides a comparison of printer support for color spaces

	Printer Support for Color Spaces			
Color Space	**HP Color LaserJet 8550 and 8500**	**HP Color LaserJet 4500 and 4550**	**HP Color LaserJet 5/5M**	**DJ1200C**
RGB	Yes	Yes	Yes	Yes
CMY	Yes	Yes	Yes	No
cRGB	sRGB	sRGB	Yes	No
CIE L*a*b	No	No	Yes	No
Luminance-Chrominance spaces	No	No	Yes	No
Standard RGB	Yes	Yes	Yes	No

Unsupported color spaces will be ignored. Therefore, rendering will proceed in the color space in effect when the command to switch to the unsupported color space was encountered.

Resetting the Printer

Resetting the printer (*EscE*) performs the following color related functions:

- Overwrites the active palette with a non-programmable black and white palette.
- Deletes all palettes on the palette stack and in the palette store (palettes created with IDs).
- Resets the palette ID to 0.
- Sets the Foreground color to black.
- Deletes all patterns.

Palettes

- Palettes are made up of a collection of color specifications which can be selected using index numbers. Each table entry within a palette associates an index number with three primary RGB or CMY color components that represent a specific color or shade of a color.
- All palettes have a unique ID.
- Default palettes are created by all PCL color modes.
- Multiple palettes can exist in the system via the palette ID and palette stack mechanisms, but only one palette at a time can be active.
- A palette created in the PCL context remains active and unchanged when switching to HP-GL/2.
- A palette created in HP-GL/2 remains active and unchanged when switching to PCL.
- For HP-GL/2 purposes only, a pen width is also associated with each color table entry.
- Performing a reset or entering PJL overwrites the active palette with the default black and white palette.
- Creating a new palette destroys the active palette. A new palette is created under the following circumstances:
    - Power-up and *EscE* (default black and white palette)
    - Simple Color *(Esc*r#*U)
    - Configure Image Data *(Esc*v#*W)
    - HP-GL/2 Initialize *(IN)*
- The active palette can be saved by pushing it onto the palette stack with the Push/Pop Palette command *(Esc*p#*P) or by associating it with an ID number by creating a palette (*Esc&p#i6C*).
- Popping a palette from the stack overwrites the currently active palette and becomes the new active palette.
- Saving color palettes with Palette IDs (*Esc&p#i6C*) versus pushing and popping palettes (*Esc*p#P*) provides better control over palettes, reduces the I/O stream, and eliminates the need to reconfigure palettes that are destroyed through push commands.
- Only the *IN* command (HP-GL/2 Imaging color mode) or *Esc*v#W* (Configuration Image Data command) can create a modifiable palette. Palettes can not be modified in the Black and White or Simple Color modes.
- In non-raster mode, the current pixel contains all the available colors.
- In raster mode, indexed color selection uses the palette, but direct selection (Direct by Plane or Direct by Pixel) does not.

Simple Color Palettes

- The Simple Color command *(Esc*r#*U) provides a quick way to select colors from a fixed, non-programmable palette.
- The Simple Color command *(Esc*r#*U) overwrites the currently active palette with a fixed palette.
- A Simple color palette cannot be modified.

- 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.
- In the Simple color mode, the pixel encoding mode reverts to "index by plane."
- 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.

CID Color Palettes (PCL Imaging Color Mode)

- The Configure Image Data (CID) command creates a palette based upon the parameters in its data field.
- CID-created palettes are programmable: any entry can be reassigned a different color by PCL *(Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#*I) or HP-GL/2 *(CR, PC, N*P) commands.
- Default palettes vary by color space.
- The black and white references specified by the CID command have no effect on the default palettes. However, when a CID palette entry is reprogrammed with a different color, the black and white references are used to specify the primary components of the new color.
- The HP Color LaserJet 4500, 4550, 8500, and 8550 printers only support the short form of the CID command. The long form is not supported.
- The HP Color LaserJet 4500 printer does not support less than 8 bits per index (Indexed by Pixel) with the CID command.

HP-GL/2 Palettes

- A default PCL Device RGB palette is different from an HP-GL/2 default palette.
- 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 *N*P, *P*C, *C*R).
- The Initialize (*IN*) command will always establish the 8-pen palette.

Saving Palettes

- Saving a palette to an ID (creating a palette) enables you to save color specifications and rendering using the *Esc&p#i6C* and then retrieve it later by selecting its ID (*Esc&p#S*). This places a copy of the palette in the palette store which can be retrieved multiple times within the PCL program.
- The Push/Pop Palette command *(Esc*p#*P) can save (push) the current palette and then restore (pop) it. To restore the palette, you must pop it from the stack in the correct sequence. The last palette pushed on the stack is the first palette popped from the stack. Once a palette is popped from the stack, it is permanently removed from the stack.
- 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 previously active palette, which is overwritten.
- Saving a palette saves:
    - Color definitions for each palette entry
    - Pen widths (the palette is also used in HP-GL/2)
    - Color space specification

  - Black and white references
  - Render algorithm
  - Number of bits per index
  - Pixel encoding mode
  - Monochrome print mode
  - Number of bits per primary.
- Saving a palette does **not** save the current Foreground color selection.

Push / Pop Palette Esc*p#P

- Pushes or pops the palette from the palette stack.
  - Range = 0, 1 (invalid configurations are ignored).
  - Default = 0
- A value of 0 pushes (saves) a copy of the active palette, which is unaffected.
- A value of 1 pops (restores) the most recently pushed palette and overwrites the currently active palette; the popped palette becomes the active palette. The newly popped palette then will be associated with the current *palette select ID* (a value stored as part of the modified print environment). The *palette select ID* is the value used with the last Select Palette command and the palette ID of the overwritten and now previously active palette. As with any stack, the last item pushed is the first item popped.
- Stack depth is limited by memory. Attempts to push with insufficient memory causes an out-of-memory error. Attempts to pop from an empty stack are ignored.
- Macros can push and pop palettes. A palette that was popped in an executed macro - but not a called or overlaid macro - remains in effect at the end of the macro.
- *EscE* empties the palette stack and overwrites the active palette with a non-programmable black and white palette. Exiting to PJL causes an *Esc*E.
- The HP-GL/2 commands, *IN* and *D*F, have no effect on the palette stack, but they do destroy the active palette and replace it with the default HP-GL/2 palette.
- The Palette ID *(Esc&p#*I) and Palette Control *(Esc&p#*C) commands have no effect on palettes saved on the stack.

Palette Management by ID

- All palettes have a unique identification number (ID). 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. Multiple palettes can exist in two areas: the palette *stack* and the palette *stor*e. The stack holds palettes that are pushed via a Push/Pop Palette command; the store holds palettes having palette IDs.
- 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. 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.
- 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*

- Selects a new active palette by ID. The previous active palette is unchanged.
  - Value(#) = Palette ID number
  - Default = 0
  - Range = 0 to 32767 (command is ignored for unsupported values)
- 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 (part of the graphics state).
- 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 *I*N, the created palette overwrites the active palette and is assigned the current *palette select ID*, which is unchanged.
- A palette popped from the stack overwrites the active palette and is assigned the current *palette select ID*, which is unchanged.
- *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* value as follows:
  - Calling or Overlaying — saves the ID value and a copy of the active palette. Upon macro exit, the restored palette again becomes the active palette with the restored ID. An existing palette with this ID is deleted.
  - Executing — Does not save the ID value or the active palette: changes remain in effect.

Palette Control ID Esc&p#I

- Specifies the ID to be used by the Palette Control command.
  - Value(#) = Palette ID number.
  - Default = 0
  - Range = 0 to 32767 (command ignores unsupported values)

- The ID specified by this command is saved as the *palette control ID* (in addition to the *palette select ID*) value in the current modified print environment and is used by the Palette Control command *(Esc&p#*C).
- *EscE* or power-up resets the *palette control ID* to 0, which is then the default black and white palette ID.
- Macros affect the *palette control ID* value as follows:
    - Calling — saves the value and restores it at exit.
    - Executing — Does not save the value: changes remain in effect at exit.
    - Overlaying — copies the value before resetting to 0, and restores it at exit.

Palette Control Esc&p#C

- Provides a mechanism for copying and deleting palettes.
    - Range = 0,1,2,6 (command is ignored for unsupported values)
    - Default = 0
- 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&p53i2*C. 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.
    - When the active palette is replaced by the default black and white palette, the graphics state associated with the previously active 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&p14i6*C. 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. To become the active palette, the Palette Select command must be executed. The Palette Control 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.
- The use of the Palette Control command within macros should be discouraged because the palette list is not saved upon macro entry.

Programming Palettes

Except for the default black and white palette or the Simple Color palette *(Esc*r#*U), palette entries can be modified. The three primary components of a color are specified and the resulting color is assigned to the palette entry indicated by *Esc*v#I* (Assign Color Index command).

In the discussion below, *components* refer to the color space primaries. For example, if the current color space is RGB, component 1 indicates the amount of Red (R), component 2 indicates the amount of Green (G), and component 3 indicates the amount of Blue (B).

Color Component One Esc*v#A

Specifies the first primary of the palette entry designated by Assign Color Index *(Esc*v#*I).

- Value(#) = First Component
- Default = 0
- Range = -32767.0000 to 32767.0000 (up to 4 decimal places; command is ignored for invalid configurations)

The Assign Color Index *(Esc*v#*I) command actually applies this value and then resets it to 0.

Color Component Two Esc*v#B

Specifies the second primary of the palette entry designated by Assign Color Index *(Esc*v#*I).

- Value(#) = Second Component
- Default = 0
- Range = -32767.0000 to 32767.0000 (up to 4 decimal places; command is ignored for invalid configurations)

The Assign Color Index *(Esc*v#*I) command actually applies this value and then resets it to 0.

Color Component Three Esc* v # C

Specifies the third primary of the palette entry designated by Assign Color Index *(Esc*v#*I).

- Value(#) = Third Component
- Default = 0
- Range = -32767.0000 to 32767.0000 (up to 4 decimal places; command is ignored for invalid configurations)

The Assign Color Index *(Esc*v#*I) command actually applies this value and then resets it 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*(number of bits per index - 1) -- no assignment for out-of-range values

This command resets the color components to 0 after assignment. No assignment is made if the specified index number is greater than the palette size; but the three color components are still set to 0.

Color Modes

Four color modes are supported:

- Black and White mode
- Simple Color mode
- PCL Imaging mode
- HP-GL/2 Imaging mode

All four color modes create a palette.

Black and White Mode

- The Black and White mode is the default mode.
- The Black and White mode has an unmodifiable, 2-pin palette where index 0 is white and index 1 is black.
- This mode is compatible with monochrome PCL 5 printers.
- The Black and White mode is also selectable using the Simple Color command (*Esc*r1U*).

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 pixel encoding mode is indexed planar.
- The Simple Color command *Esc*r#U* creates a fixed-size palette, whose color specification cannot be modified.
- A value field of 1 creates a 2-entry black and white default HP LaserJet palette, where index 0 is white and index 1 is black.
- 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), where index 0 is black and index 7 is white.
- A value field of -3 creates an 8-entry palette in Device CMY color space, where index 0 is white and index 7 is black.

Single Plane Esc*r1U	3-Plane RGB Esc*r3U	3-Plane CMY Esc*r-3U	Color
1	**0**	**7**	**Black**
	1	**6**	**Red**
	2	**5**	**Green**
	3	**4**	**Yellow**
	4	**3**	**Blue**
	5	**2**	**Magenta**
	6	**1**	**Cyan**
0	**7**	**0**	**White**

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

- The Simple color mode is not recommended for color raster data because it is inefficient. In this mode RGB and CMY raster must be sent by plane and row (indexed by plane) resulting in additional data processing and larger files.

PCL Imaging Mode (CID Command)

- Palettes are created in the PCL Imaging mode using the Configure Image Data (CID) command which creates a palette based upon the parameters in its data field.
- The PCL Imaging mode provides up to 24 bits per pixel (8 bits per primary).
- CID-created palettes are programmable. The pixel encoding mode, bits per pixel, bits per primary, and color palette are all programmable. In addition, any entry can be reassigned a different color by PCL *(Esc*v#A, Esc*v#B, Esc*v#C, Esc*v#*I) or HP-GL/2 *(CR, PC, N*P) commands.
- The black and white references specified by the CID command have no effect on the default palettes. However, when a CID palette entry is reprogrammed with a different color, the black and white references are used to specify the primary components of the new color.
- The HP Color LaserJet 4500, 4550, 8500, and 8550 printers only support the short form of the CID command. The long form is not supported.
- The HP Color LaserJet 4500 printer does not support less than 8 bits per index (Indexed by Pixel) with the CID command.
- Default palettes vary by color space.

Configure Image Data (CID) Command

Esc*v6W b0 b1 b2 b3 b4 b5

Where:

- 6 = the number of bytes following the command
- b0 = byte 0 = the color space
- b1 = byte 1 = the Pixel Encoding mode
- b2 = byte 2 = the number of bits per index or palette size
- b3 = byte 3 = the number of bits in the color component
- b4 = byte 4 = the number of bits in the color component
- b5 = byte 5 = the number of bits in the color component

Bytes 0 through 5 must contain binary data, not ASCII.

CID Byte 0, Color Space

The HP Color LaserJet 4500, 4550, 8500, and 8550 printers support the following color spaces for the CID command:

- Byte 0 = 0 = Device Dependent RGB (Red, Green, Blue).
- Byte 0 = 1 = Device Dependent CMY (Cyan, Yellow, Magenta).
- Byte 0 = 2 = Standard RGB (sRGB).

CID Byte 1, Pixel Encoding Mode

The Pixel Encoding mode designates the format of any subsequent raster images. There are two major formats for the Pixel Encoding mode: by plane or by pixel.

- Byte 1 = 0 = Indexed by Plane
- Byte 1 = 1 = Indexed by Pixel
- Byte 1 = 2 = Direct by Plane
- Byte 1 = 2 = Direct by Pixel

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. A pixel is not fully defined until it has received all the planes for that row.

For data encoded Indexed by Plane, the planes in a row form index numbers that defines a pixel by selecting a palette entry. For example, an 8-entry palette requires 3 planes. The highlighted bits below compose the index of the color of 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...**

For data encoded Direct by Plane, the planes in a row directly specify a pixel which is composed of three, one-bit components. For example, Device RGB requires 3 planes. The highlighted bits below compose the color of the fourth pixel in the first row.

Esc*b#V	**row 1**	**plane 1**	**r r r r r r r r ...**
Esc*b#V		**plane 2**	**g g g g g g g g ...**
Esc*b#W		**plane 3**	**b b b b b b b b ...**
Esc*b#V	**row 2**	**plane 1**	**r r r r r r r r ...**

When encoded by pixel, each pixel is fully specified before any bits are sent for the next pixel and the bits for each pixel form a palette index number. For example, for data encoded Indexed by Pixel -- for a 16-color palette (4 bits per index), every group of four bits in the data stream defines a pixel. The highlighted (c4...c1) group below defines the palette index value for the second pixel in the first row.

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

For data encoded Direct by Pixel, data is sent pixel by pixel, as three eight-bit components. Each pixel specifies each color component. For the Device RGB color space, pixel 1 of row 2 is highlighted.

Esc*b#W	**row 1:**	**r7-r0 g7-g0 b7-b0 ...**
Esc*b#W	**row 2:**	**r7-r0 g7-g0 b7-b0 ...**
Esc*b#W	**row 3:**	**r7-r0 g7-g0 b7-b0 ...**

Direct by Pixel is the recommended format since it provides the most efficient raster processing.

CID Byte 2, Bits per Index (Palette Size)

- The bits per pixel determine the size of the palette or number of bits used to hold a pixel. Basically, this determines the number of colors that can be displayed at one time.
- The palette size is two raised to the power of n (2^n) where "n" is the bits per index value for example, a 256 entry palette requires 8 bits per component -- bits per index = 8 where $2^8 = 256$.

Bits per Pixel	Number of Colors
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

- For Indexed by plane, this value determines the number of planes required per row of data.
- For Indexed by Pixel, this value determines how to interpret the byte-ordered row transfers.
- For Direct by Plane or Direct by Pixel, this value does not apply to the raster format.

Pixel Encoding Mode	Restrictions
Indexed by Plane (Default)	1, 2, 3, 4, 5, 6, 7, or 8 bits per index
Indexed by Pixel *	1, 2, 4, or 8 bits per index
Direct by Plane	1 bit per primary (3-bit color) -- RGB or CMY only
Direct by Pixel (recommended format for raster)	8 bits per primary (24-bit color)

* The HP Color LaserJet 4500 printer does not support less than 8 bits per index with the CID command.

CID Bytes 3 - 5, Bits per Color Component

- These bits determine the number of bits in the color component for each primary.
- These bytes are ignored for Indexed by Plane and Index by Pixel pixel encoding modes.
- For Direct by Plane, these bytes must be set to one bit per component.
- For Direct by Pixel, these bytes must be set to eight bits per component.

CID Device RGB and sRGB Color Palettes (Default)

Bits/Index = 1	Bits/Index = 2	Bits/Index = 3 to 8	
Index	**Index**	**Index**	**Color**
1	**0**	**0**	**Black**
	1	**1**	**Red**
	2	**2**	**Green**
		3	**Yellow**
		4	**Blue**
		5	**Magenta**
		6	**Cyan**
0	**3**	**7**	**White**
		n > 7	**Black**

CID Device CMY Color Palettes (Default)

Bits/Index = 1	Bits/Index = 2	Bits/Index = 3 to 8	
Index	**Index**	**Index**	**Color**
1	**0**	**0**	**White**
	1	**1**	**Cyan**
	2	**2**	**Magenta**
		3	**Blue**
		4	**Yellow**
		5	**Green**
		6	**Red**
0	**3**	**7**	**Black**
		n > 7	**Black**

HP-GL/2 Imaging Mode

- The HP-GL/2 Imaging mode provides a way of using vector commands in printing documents.
- When transferring between PCL and HP-GL/2, you can use the same palette.
- In HP-GL/2, the Initialize (*IN*) command starts color imaging and establishes the 8-pen palette.
- 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 *N*P, *P*C, *CR*).
- With HP-GL/2, the pixel encoding mode is set to "Index by Plane," and the bits per index is set to three.
- A default PCL Device RGB palette is different from an HP-GL/2 default palette.

For more information on HP-GL/2, see the *PCL 5 Printer Language Technical Reference Manual*.

Raster Images

PCL Raster Features

Raster processing requirements for the HP Color LaserJet 4500, 4550, 8500, and 8550 printer include the following:

1. Accepts all PCL raster commands.
2. Operates in a limited and fragmented memory environment.
3. Has a firmware architecture that will accept an unordered raster stream and perform limited or basic raster healing.
4. Is independent of feed direction and device orientation.
5. Supports both Monochrome and Color Devices with the appropriate compilation switches.
6. Uses the Graphics Engine[GEIF] and the Common Object Marking Architecture [COMA] to:
    - Manage formatting of raster blocks into internal format.
    - Perform scaling, rotation, translation, and resolution changes.
    - Perform clipping of data outside the logical page.
    - Interpret the raster's color specification into device color.
    - Accept all resolutions (75 to 600) and perform operations to transform the image to device resolution.
    - Do raster scaling.

Well-behaved Raster

Well-behaved raster is a sequence of PCL raster commands that describe a complete image, start at the top left hand corner, and proceed to the bottom of the image.

With Ill-behaved Raster, on the other hand, problems occur when the raster blocks are not transferred in sequence (blocks 1, 4, 3, and 2 are transferred in that order). Further complications arise if multiple sequences are used to transfer portions of the same image. In situations like this, the PCL system could be fooled into thinking that the raster is four images instead of four parts of the same image. The creation of spurious edges and the application of nearest neighbor half-toning or scaling algorithms such as error-diffusion or linear interpolation compound the problem. This situation creates visible lines within the image and other errors called stitching artifacts that are created when the blocks of raster data are merged into a single raster image.

Healing Raster

Various techniques or heuristics are applied to heal the disjoint raster blocks into a single raster block, thereby avoiding stitching problems. Delaying the commitment of the raster image to the graphics engine performs the healing function.

Committing the Image to the Graphics Engine

The raster system will delay the ending of an image so that disjoint raster transfers can be treated as single image. The following list shows the conditions when the raster must be closed:

- Just before the end page call is made to the graphics engine, but after any macro overlay is complete.
- Before the PCL orientation changes.
- Before logical page width or length are changed.
- Before the registration changes.
- Before the logical page home position changes.
- When the source or pattern transparency is opaque and we are ending a raster graphics section.
- Before a change in the current pattern.
- Before a change in the tile reference point for the current pattern.
- Before a change in transparency modes.
- Before a change in the height and width of the destination raster image.
- When a CID command changes the raster data formats or color definitions.
- When palette commands change color definitions.
- When a delete, push/pop, or select palette operation occurs.

Color PCL Raster

Color raster images can be composed (conceptually) of three color planes of data, which describe the image. How this data is transferred to the printer is set by the pixel encoding mode portion of the Configure Image Data command.

Color Specifications: Indexed or Direct

Each bit in the raster image (whether in planar or pixel mode) has a color attribute. This attribute can be either an index into the color palette or a direct color description in any of the color spaces supported by the device.

Filling the Raster Area

The raster area is defined as a bounded raster picture and calls for the PCL raster system to "zero-fill" missing and incomplete rows, as well as clipping data outside the raster area.

In a monochrome device, zero is the absence of data, which is identical to white -- nothing is printed. In a color device using a monochrome or CMY color space, zero represents white. However, if the RGB palette is chosen, or the user programs a palette, then something other than white (perhaps black) will be printed when zero filling is performed.

Missing Rows

When the Raster Y Offset command is used in place of a series of zero length Transfer Row commands, then zero filling to represent null data cannot be distinguished from rows where the color specification or index is zero. Therefore, colored (non-white) areas may appear when the color space is not monochrome or CMY.

Incomplete Rows

This situation occurs when the data transferred does not fill to the raster source width. As with missing rows, distinguishing null or absent data from data with a color specification or index of zero is very difficult. Therefore, colored (non-white) areas may appear when the color space is not monochrome or CMY.

Use-Case Analysis

Case 1: well-formed raster with a single image on the page

This case deals with a large raster image that takes most of the logical page. There may be text before or after the image, but none to the left or right. Since the raster is well formed or well behaved, the image attributes, destination size, and height for example, have been set by the commands in the raster prologue. This implies that the raster image attributes have been completely specified without recourse to default values or values from a previous image. Furthermore, the raster data is transferred smoothly as a contiguous image either top to bottom, or bottom to top. Partial rows might be encountered and will be filled with a zero value.

- Monochrome raster -- The default mode is Black and White where index 0 in the palette is white or effectively transparent.
- Color Raster -- The three (supported) Simple Color palettes: are monochrome, RGB, and CMY. Two of which have white (or transparent) for index 0, and the third, RGB, has black for index 0. A simple zero filling for incomplete raster lines will give either transparent or black depending on the selected palette. The Configure Image Data command further complicates matters by defining palettes where any color can be assigned to index 0.

Case 2: Well formed raster with multiple, small images on the page

This case deals with mixed images and text -- text wrapped around images as well as interspersed in the text. Two separate images, which were aligned in the page, were transmitted as a single row with transparent fill between them so that the text would show.

- Monochrome Raster -- In this case, a zero fill between the two images works, since, zero is white.
- Color images -- Color images cannot have a zero fill, since in the RGB color space (and potentially user-defined palettes) this would place color between the images.

Case 3: Lazy raster with a single image on a page

This is the case of ill-behaved raster -- no raster prologue, just transfer commands. Here, the current values or system defaults are applied to the image. Since there is no clue what the dimensions of the image are, it must be buffered until one of the closing conditions is encountered. Furthermore, there's no guarantee that the rows will arrive in sequential order. Buffering the data will order it for the eventual transfer to the graphics engine. Since neither the source or destination sizes are given, the size of the transferred image will be used and no scaling will be required.

- Monochrome Raster -- The HP LaserJet 5Si firmware code base raster manager assumed that the width of the image was the size of the first row transferred.
- Color Raster -- The Logical page width is the default width of the image. Therefore, the rows are filled with index zero. This may give unintended results when accepting input from an HP Monochrome LaserJet driver and a palette other than the monochrome and CMY palettes are used.

Case 4: Lazy raster with multiple, small images on a page

As in case 3, the input must be buffered to determine the image size. Furthermore, the problems discussed in Case 2 also apply.

Case 5: Raster Prologue used with unordered transfers

This is a modification of the middle case above. The image attributes are stated, but the rows are not transferred in sequential order. There are large jumps on the positive and negative Y direction. Here we can supply fill values that are discarded if the area is later transferred. This does require buffering for coherency before transferring the image to the graphics engine.

Raster Graphics Commands

These commands deal with raster (bitmap) graphics. The commands are grouped as follows:

- Raster Mode
- Raster Data Transfer
- Raster Compression
- Raster Scaling

Raster Mode

Raster mode is a restricted state, which can be entered and exited either explicitly or implicitly. The implicit enter and exit operations are not recommend, but are supported.

- Raster mode is entered explicitly with the Start Raster command (*Esc*r#A*) or implicitly with the Transfer Raster command (*Esc*b#V, Esc*b#W*) and raster Y offset command.
- Raster mode is exited explicitly by an End Raster command (*Esc*rC*) or implicitly by a non-raster command.

Command	Sequence	Description/Range
Raster Presentation Mode	Esc*r#F	0 = raster row will be printed in the positive X-direction of the PCL coordinate system (logical page) 3 = raster graphics will be printed along the width of the physical page, regardless of the logical page orientation.
Raster Resolution	Esc*t#R	Defines the X and Y dimensions of a raster pixel -- the number of dots or pixels per inch. Values = 75, 100, 150, 200, 300, 600 150 is recommended for good print quality at a relatively small image size.
Source Raster Height *	Esc*r#T	Specifies the height of the raster area in raster rows. Range = 0 to Logical Page Bound Printer zero fills if too few rows are sent and clips if too many rows are sent.
Source Raster Width *	Esc*r#S	Specifies the width of the raster area in pixels. Range = 0 to Logical Page Right Printer zero fills if too few bytes are sent and clips if too many bytes are sent.

Start Raster	Esc*r#A	Identifies the beginning of the raster data 0 = sets the graphics margin to the default left graphics margin (X-Position 0) 1 = sets the graphics margin to the current cursor position (current X-Position) 2 = raster scaling on, start at the left boundary 3 = raster scaling on, start at the current active position
Raster Y Offset	Esc*b#Y	Moves the cursor vertically the specified number of raster lines from the current raster position. Range = 0-32767 Printer zero fills the offset area.
End Raster Graphics	Esc*rC	Signifies the end of a raster data transfer. A newer version of the End Raster Graphics command which performs two additional operations: resets the compression mode to 0 (zero) and defaults the left graphics margin to 0 (zero).
End Raster Graphics	Esc*rB	Older version of the End Raster command for the HP LaserJet III and IIID printers.

* Specifying the Height and Width of the raster improves printer memory efficiency.

Raster Data Transfer

The raster transfer commands define how many bytes are to be interpreted as binary raster data, rather than as ASCII data. The data may be encoded either by plane or by row.

Command	Sequence	Range
Transfer Raster by Plane	Esc*b#V[data]	0-32767
Transfer Raster by Row/Block	Esc*b#W[data]	0-32767

Upon completion of the Esc*b#W command, the cursor position is at the beginning of the next raster row at the left raster margin.

Raster Compression

Compressed data formats can improve data transfer throughput and efficiency. There are several different compression modes, which determine how the raster data in the transfer raster commands is interpreted.

Command	Sequence	Range
Compression Method	Esc*b#M	0 – unencoded
		1 -- Run-length encoding
		2 -- TIFF rev 4.0 encoding
		3 -- Delta row encoding
		5 -- Adaptive (block-based) for monochrome only

Raster Scaling

Raster scaling provides the ability to enlarge or reduce raster graphic images. The Start Raster command (Esc*r#A) with a value of 2 or 3 turns on the scale mode. Scaling is independent of the device resolution. The scaling factor is implicitly carried in the destination size of the raster image.

Command	Sequence	Description/Range
Destination Raster Width	Esc*t#H	Defines the width in decipoints of the destination raster image. Range = 0-32767
Destination Raster Height	Esc*t#V	Defines the height in decipoints of the destination raster image. Range = 0-32767

Raster Printing Command Sequence

Listed below is the recommended raster command sequence for sending raster images.

Raster Presentation
Raster Resolution
Raster Height
Raster Width
Start Raster Graphics
Y Offset
Raster Compression
Transfer Raster data

...

Transfer Raster Data
Y Offset
Transfer Raster Data

...

Y Offset
Raster Compression
Transfer Raster Data

...

Raster Compression
Transfer Raster Data
End Raster Graphics

Foreground Color

- All PCL marking entities except color user-defined patterns (created via the Download Pattern format 1 command) utilize foreground color, which is selected by Esc*v#S from the current palette.
- Color raster mixes with foreground color, so foreground color should be set to black when sending color raster images.
- Foreground color does not affect color user-defined patterns (format 1 download patterns).
- HP-GL/2 page marking primitives use the selected pen instead of PCL foreground color.

Caution

Foreground color should be set to black when sending raster data. Failing to do this may result in corrupted colors within the raster images. For example, setting foreground color to red will result in a red component being mixed with each of the color components of the raster image. A green component may appear yellow, a blue component may appear magenta, and so on.

Foreground Color Esc*v#S

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

- Value(#) = Index Number Into Current Palette
- Default = 0
- Range = 0 to 2 current palette size -1

Out-of-range values are mapped into a new index via modulo(palette size). For example, if the current palette size is 8, and the selected index for the foreground color command is 10, the index is mapped into the index of 2 (i.e., 10 modulo 8).

Foreground color affects the following PCL page marking primitives:

- Text characters
- Single-color or monochrome patterned rectangular area fills (rules)
- Single-color or monochrome patterns (except HP-GL/2)
- Vector images.
- Raster images (all three components -- RGB or CMY)

The following are not affected:

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

Note

Foreground color interacts with color raster images. To avoid undesired interactions, select a black foreground color.

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

- Active palette
- Configure Image Data command
- Render Algorithm

Monochrome Print Mode *(Esc&b1M)* immediately maps foreground color to its NTSC gray equivalent and *(Esc&b0M)* immediately returns foreground color to its full color.

Render Algorithms

The render algorithm command provides a way to modify images based on a dither cell concept. This determines how specified colors are rendered as dots on the printed page.

Command	**Sequence**	**Range**
Render Algorithm	Esc*t#J	0 = Continuous tone detail (high lpi)
		3 = Device-best dither (Default)
		15 = Continuous tone smooth (high lpi)
		18 = Continuous tone basic (low lpi)

- The detail selection is for text and images containing a lot of small picture elements with very little smooth tonal areas.
- The smooth selection is for photos or images that contain large areas of flat, smooth tonal regions such as presentation backgrounds.

Monochrome Print Mode/Finish Mode

Printed output can be modified as desired. Both the Monochrome Print Mode and the Finish Mode commands apply only to an entire PCL job. Both commands must be sent to the printer before any printable data. The commands cannot be changed after printable data has been received except by a Reset command which returns the printer to a matte finish in the case of the Finish Mode command and current rendering mode in the case of the Monochrome Print Mode command. If the command is received after printable data, then the command is ignored.

Command	**Sequence**	**Description/Range**
Monochrome Print Mode	Esc&b#M	0 = Print in the mixed render algorithm mode 1 = Print using the gray-scale equivalent
Finish Mode	Esc&b#F	0 = Matte finish (default) 1 = Glossy finish

Driver Configuration

The Driver Configuration command specifies the color treatment applied to each color specification.

Command	**Sequence**	**Range**
Driver Configuration	Esc*o#W<device><function><value>	# = number of bytes to follow
		<device> = 6 for HP Color LaserJet printers <device> = 8 for HP Color LaserJet 4500 and 4550 printers.
		<function> = 4 for select color treatment
		<value> = 3 for Vivid <value> = 6 for Screen Match

<device>, <function>, and <value> must be entered as ASCII-coded decimal

- Vivid adds color saturation to an image and provides access to the full gamut of the printer, but may impact color matching.
- Screen Match is an sRGB mode which provides a good match between the monitor and the printed document.

The Color Print Model

The PCL print model allows images and characters to be filled with color and patterns. Images include raster graphics, rectangular area fills, font characters, and HP-GL/2 objects. The commands are grouped as follows:

- Logical Operations
- Transparency Modes
- Pixel Placement
- Patterns
- User-Defined Patterns
- Rectangular Area Fills (Rules)

Logical Operations

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

Command	**Sequence**	**Range**
Logical Operation	Esc*l#O	0-255

For more information, see chapter 5 of the *PCL 5 Color Technical Reference Manual*.

Transparency Modes

Transparency modes define how the white source and pattern affect the destination. White source and pattern pixels are either transparent and have no effect on the destination, or they are opaque and appear white on the destination.

The default PCL print model is that source and pattern transparency modes are transparent, and the logical operation is Texture or Source.

Command	Sequence	Description/Range
Source Transparency Mode	Esc*v#N	0 = Transparent -- white regions of the source image are not copied to the destination. 1 = Opaque -- white regions of the source image are applied to the destination.
Pattern Transparency Mode	Esc*v#O	0 = Transparent -- white regions of the pattern image are not copied to the destination. 1 = Opaque -- white regions of the pattern are applied directly to the destination.

Pixel Placement

By default, the printer places pixels at the intersections of a device-dependent grid that covers the printable area on a page. There are two pixel placement models:

- Grid intersection which places pixels at the intersection points of the grid.
- Grid centered, which places pixels at the center of the squares, formed by the grid.

Command	Sequence	Range
Pixel Placement	Esc*l#R	0 = Grid intersection 1 = Grid centered

Patterns

The commands for applying patterns to text and raster images are shown in the following table. For selecting or changing the current pattern, the Select Current Pattern and the Pattern ID commands work together. Sending the Current Pattern command alone does not change the current pattern. You must send the Pattern ID command first.

Command	Sequence	Description/Range
Pattern ID	Esc*c#G	0 - 6 for selecting cross-hatch patterns 1 - 100 for selecting shaded patterns
Current Pattern	Esc*v#T	0 = Solid black (default) 1 = Solid white 2 = Shading pattern 3 = Cross-hatch pattern 4 = User-defined pattern

User-Defined Patterns

User defined patterns, which are downloaded to the printer are shown in the following table.

Command	Sequence	Description/Range
User-Defined Pattern *	Esc*c#W[data]	Provides a means for downloading the binary user-defined pattern data. 0 - 32767 data bytes (The HP Color LaserJet 8500 printer extends this range to 65535)
Pattern Reference Point	Esc*p#R	Causes the printer to tile patterns with respect to the current cursor position. 0 = Rotate patterns with the print direction 1 = Keep patterns fixed.
Pattern Control	Esc*c#Q	Provides a means for manipulating user-defined patterns. 0 = Delete all temporary and permanent patterns. 1 = Delete all temporary patterns. 2 = Delete pattern with the last ID # specified. 4 = Make the pattern of the last ID specified temporary. 5 = Make the pattern of the last ID specified permanent.

* In addition to the number of data bytes, there are eight bytes of pattern descriptor (header) information included in this data. For header information, see chapter 5 of the *PCL 5 Color Technical Reference Manual*.

Rectangle Area Fills (Rules)

- Rules are a special case of source images.
- The source transparency mode has no effect, since the rectangular area is conceptually viewed as an all 1's source.
- Rules may be filled using patterns or textures.
- The Pattern ID command (*Esc*c#G*) selects the pattern.

Command	Sequence	Description/Range
Fill Rectangular Area	Esc*c#P	0 = Solid black or foreground color 1 = Solid white fill 2 = Shaded fill 3 = Cross-hatch fill 4 = User-defined pattern fill 5 = Current pattern fill
Horizontal Rectangle Size (decipoints)	Esc*c#H	0-32767
Horizontal Rectangle Size (PCL Units)	Esc*c#A	0-32767
Vertical Rectangle Size (decipoints)	Esc*c#V	0-32767
Vertical Rectangle Size (PCL Units)	Esc*c#B	0-32767

Vector Graphics

PCL 5 includes most of the HP-GL/2 command set. This section describes commands that enter and exit the HP-GL/2 context and manage the scaling of HP-GL/2 graphics to the PCL page.

When entering the HP-GL/2 context from PCL, the HP-GL/2 graphic exists only in the context of a PCL picture frame. The picture frame is the destination rectangle on the logical page into which the HP-GL/2 graphic is placed. For more information on HP-GL/2 commands, see the *PCL 5 Printer language Technical Reference Manual*.

PCL, HP-GL/2 Context Switching

These commands allow the device to switch between PCL and HP-GL/2.

Command	Sequence	Range
Enter PCL mode	Esc%#A	0 = Use PCL CAP 1 = Use GL pen position for CAP
Enter HP-GL/2 mode	Esc%#B	0 = Use previous or default GL pen position 1 = Use PCL CAP

Picture Presentation Commands

The default picture frame normally defines the plot's destination; but picture presentation commands can be used to modify the size and position of the picture frame, as well as the plot size with respect to the picture frame.

Command	Sequence	Range
Picture Frame Anchor Point	Esc*c#T	0
Picture Frame Horizontal Size (Decipoint)	Esc*c#X	0-32767
Picture Frame Vertical Size (Decipoints)	Esc*c#Y	0-32767
Plot Horizontal Size	Esc*c#K	0-32767
Plot Vertical Size	Esc*c#L	0-32767

Using Macros with Color

- Avoid creating or modifying palettes within macros.
    - A palette that was popped in an executed macro remains in effect at the end of the macro.
    - Executing a macro does not save the ID value or the active palette -- changes made in the macro remain in effect at the end of the macro.
    - Calling or overlaying a macro saves the ID value and a copy of the active palette. Upon exiting the macro, the saved palette is restored along with its ID. If a palette was created within the macro with the same ID, it is deleted.
- Avoid controlling palettes within macros.
    - Deleting all palettes, palettes on the stack, the current palette, or all palettes except those on the stack within a macro can have adverse effects when the macro is exited. The result could range from accidentally deleting a needed palette to replacing a programmable palette with a black and white non-programmable palette.
- The use of the Palette Control command within macros should be discouraged because the palette list is not saved upon macro entry.

Asian Font Support

The following commands support Asian Font printing:

Command	Sequence
Character Text Path Direction	Esc&c#T
Text Parsing Method	Esc&t#P
Download Font	Esc)s#W[font def]

New Asian Paper Sizes Supported

The following commands support the new Asian paper sizes in the printer.

Media	**Dimensions** **[media points]**	**PCL (Esc&l#A)**	**PJL**	**PML**
16K	[558 774] 7.75 x 10.75 inches	Esc&l17A	ROC16K	eROC16K(17)
8K *	[774 1116] 10.75 x 15.5 inches	Esc&l19A	ROC8K	eROC8K(19)
JIS Executive	[612 935] 216 x 330 mm	Esc&l18A	JISEXEC	eJISExecutive(18)

* K is not supported in the HP Color LaserJet 4550 printer.

Programming Examples

The following PCL 5c programming examples are included to show you the correct flow of commands for different printing examples. Comments are interspersed with the commands to clarify what is happening in the code. The actual code is available in the PCLSample.zip file available on the web site with this document. File names for each example are listed at the beginning of the sample code. These files can be printed through the parallel port from the DOS prompt with the Copy command (Copy <file> /b LPT1).

Note

HP recommends beginning and ending all jobs with the Universal Exit Language PJL command *(Esc%-12345*X) and @PJL ENTER LANGUAGE = PCL. To focus on the PCL structure, these have not been shown in the following programming examples.

Printing with Color Text

File Name: ColorPcltxt

The following example shows how to print with colored text. Fixed Color Palettes are created using the Simple Color command Esc*r3U.

Output: This program prints 6 lines of text, each line in a different color.

```
EscE
Esc*r3U           select an 8-entry RGB Simple Color palette
Esc*v1S           sets foreground color
This is a text in Red
Esc*v2S
This is a text in Green
Esc*v3S
This is a text in Yellow
Esc*v4S
This is a text in Blue
Esc*v5S
This is a text in Magenta
Esc*v6S
This is a text in Cyan
EscE
```

Printing with Color PCL Rules

File Name: ColorPclRule

The following example shows how print with color PCL rules. Fixed Color Palettes are created using the Simple Color command.

Output: This program prints six 1-inch by 1-inch color rectangles (red, green, yellow, blue, magenta, and cyan) down the left side of the page.

```
EscE
Esc*r3U           8 entry RGB palette
Esc*v1S           sets foreground color
Esc*c300a300bP    specifies the rectangle width and height
Esc*p0x+300Y      moves cursor to PCL unit position
Esc*v2S
Esc*c300a300bP
Esc*p0x+300Y      moves cursor to PCL unit position
Esc*v3S
Esc*c300a300bP
Esc*p0x+300Y      moves cursor to PCL unit position
Esc*v4S
Esc*c300a300bP
Esc*p0x+300Y      moves cursor to PCL unit position
Esc*v5S
Esc*c300a300bP
Esc*p0x+300Y      moves cursor to PCL unit position
Esc*v6S
Esc*c300a300bP
Esc*p0x+300Y      moves cursor to PCL unit position
EscE
```

Printing with Hatch Patterns on PCL Rules

File Name: ColorPclHatch

The following example shows how to print hatch patterns on PCL rules. Fixed Color Palettes are created using the Simple Color command and rectangles are filled using the Select Current Pattern and Pattern ID commands.

Output: This program print six 2-inch by 2-inch color rectangles with each rectangle containing a different hatch pattern.

Note

For selecting or changing the current pattern, the Select Current Pattern and the Pattern ID commands work together. Sending the Current Pattern command alone does not change the current pattern. You must send the Pattern ID command first.

```
EscE
Esc*r3U           selects an 8-entry RGB Simple Color palette
Esc*v1S           sets foreground color to red
Esc*p300x300Y     moves cursor to PCL unit position (300,300)
Esc*c1G           identifies type of pattern to use (horizontal lines)
Esc*c600A         sets the rectangle width to 600 PCL units
Esc*c600B         sets the rectangle height to 600 PCL units
Esc*c3P           determines type of pattern used to fill the rectangle
Esc*p+900x+0Y     move cursor
Esc*v2S           sets foreground color to green
Esc*c2G           selects vertical lines
Esc*c600A
Esc*c600B
Esc*c3P
Esc*p300x+900Y    move cursor
Esc*v3S           sets foreground color to yellow
Esc*c3G           selects right to left angled lines
Esc*c600A
Esc*c600B
Esc*c3P
Esc*p+900x+0Y     move cursor
Esc*v4S           sets foreground color to blue
Esc*c4G           selects left to right angled lines
Esc*c600A
```

```
Esc*c600B
Esc*c3P
Esc*p300x+900Y    move cursor
Esc*v5S           sets foreground color to magenta
Esc*c5G           selects a cross-hatch pattern
Esc*c600A
Esc*c600B
Esc*c3P
Esc*p+900x+0Y     move cursor
Esc*v6S           sets foreground color to cyan
Esc*c6G           selects an angled cross-hatch pattern
Esc*c600A
Esc*c600B
Esc*c3P
EscE
```

Printing with Color Text and Hatch Patterns

File Name: ColortxtHatch

The following example shows how to print with color text and hatch patterns. Fixed Color Palettes are created using the Simple Color command Esc*r3U.

Output: This program prints 6 sets of lines of text, with each set using a different hatch pattern and in a different color.

```
EscE
.H 0A
Esc(s3H           specifies the primary pitch
.H 0A
Esc*p95x200Y      moves cursor to PCL unit position (95,200)
Esc*r3U           8 entry RGB palette
Esc*v1S           sets foreground color
Esc*c1G           specifies area fill pattern
Esc*v3T           specifies the type of pattern to be applied

Hatch Pat ID :1, Horizontal lines

Esc*v2S
Esc*c2G           specifies area fill pattern
Esc*v3T

Hatch Pat ID :2, Vertical lines

Esc*v3S
Esc*c3G           specifies area fill pattern
Esc*v3T

Hatch Pat ID :3, Diagonal lines, Lower Left Upper Right

Esc*v4S
Esc*c4G           specifies area fill pattern
Esc*v3T

Hatch Pat ID :4, Diagonal Lines, Lower Right Upper Left
```

```
Esc*v5S
Esc*c5G           specifies area fill pattern
Esc*v3T

Hatch Pat ID :5, CROSS-HATCH, Horizontal and Vertical

Esc*v6S
Esc*c6G           specifies area fill pattern
Esc*v3T

Hatch Pat ID :6, CROSS-HATCH, DIAGONAL

EscE
```

Printing with Color Text and Shading Patterns

File Name: ColortxtShade

The following example shows how to print with colored text and shading patterns. Fixed Color Palettes are created using the Simple Color command Esc*r3U.

Output: This program prints six sets of lines of text, each with different amounts of shading from 15 percent to 90 percent.

```
EscE
.H 0A
Esc(s2H           specifies the primary pitch
.H 0A
Esc*p95x200Y      moves cursor to PCL unit position (95,200)
Esc*r3U           8 entry RGB palette
Esc*v1S           sets foreground color
Esc*c15G          specifies area fill pattern
Esc*v2T           specifies the type of pattern to be applied

Shading Pattern 15%

000  OOOOOOOO

Esc*v1S
Esc*c30G          specifies area fill pattern
Esc*v2T

Shading Pattern 30%

000  OOOOOOOO

Esc*v1S
Esc*c45G          specifies area fill pattern
Esc*v2T

Shading Pattern 45%

000  OOOOOOOO
```

```
Esc*v1S
Esc*c60G          specifies area fill pattern
Esc*v2T

Shading Pattern 60%

000  OOOOOOOO

Esc*v1S
Esc*c75G          specifies area fill pattern
Esc*v2T

Shading Pattern 75%

000  OOOOOOOO

Esc*v1S
Esc*c90G          specifies area fill pattern
Esc*v2T

Shading Pattern 90%

000  OOOOOOOO

EscE
```

Printing with Color Text and User-Defined Patterns

File Name: ColortxtUDP

The following example shows how to print colored text and user-defined patterns. Fixed Color Palettes are created using the Simple Color command Esc*r3U.

Output: This program prints three sets of lines of text containing different user defined patterns.

```
EscE
.H 0A
User Defined Patterns
Esc(s2H          specifies the primary pitch
.H 0A
Esc*p95x200Y     moves cursor to PCL unit position (95,200)
Esc*r3U          8 entry RGB palette
Esc*v1S          sets foreground color
Esc*c2G          specifies area fill pattern
Esc*c40W         specifies a user-defined pattern
.H 00 00 01 00 00 10 00 10 07 E0 0F F0 18 18 30 0C
.H 60 06 C0 03 C0 03 C0 03 C0 03 C0 03 C0 03 60 06
.H 30 0C 18 18 0F E0 07 E0
Esc*v4T          specifies the type of pattern to be applied
.H 55 73 65 72
.H 44 65 66 69 6E 65 64
.H 50 61 74 74 65 72 6E
.H 43 69 72 63 6C 65
Esc*v2S
Esc*c3G          specifies area fill pattern
Esc*c38W
.H 00 00 01 00 00 10 00 10 00 80 00 C0 00 E0 00 F0
.H FF F8 FF FC FF FE FF FF FF FF FF FE FF FC FF F8
.H 00 F0 00 E0 00 C0 00 80
Esc*v4T
.H 55 73 65 72
.H 44 65 66 69 6E 65 64
.H 50 61 74 74 65 72 6E
.H 41 72 72 6F 77
Esc*v4S
```

```
Esc*c4G           specifies area fill pattern
Esc*c190W
.H 00 00 01 00 00 1A 00 32 FF FF FF 80 00 00 00 FF
.H FF FF 80 00 00 00 FF FF FF 80 00 00 00 FF FF FF
.H 80 00 00 00 FF FF FF 80 00 00 00 FF FF FF 80 00
.H 00 00 FF FF FF 80 00 00 00 FF FF FF 80 00 00 00
.H FF FF FF 80 00 00 00 FF FF FF 80 00 00 00 FF FF
.H FF 80 00 00 00 FF FF FF 80 00 00 00 FF FF FF 80
.H 00 00 00 00 00 00 7F FF FF C0 00 00 00 7F FF FF
.H C0 00 00 00 7F FF FF C0 00 00 00 7F FF FF C0 00
.H 00 00 7F FF FF C0 00 00 00 7F FF FF C0 00 00 00
.H 7F FF FF C0 00 00 00 7F FF FF C0 00 00 00 7F FF
.H FF C0 00 00 00 7F FF FF C0 00 00 00 7F FF FF C0
.H 00 00 00 7F FF FF C0 00 00 00 7F FF FF C0
Esc*v4T
.H 55 73 65 72
.H 44 65 66 69 6E 65 64
.H 50 61 74 74 65 72 6E
.H 43 68 65 63 6B 65 72
.H 42 6F 61 72 64
EscE
```

Printing with Color Raster

File Name: ColorBar

This sample program processes a simple color raster image using the PCL Imaging mode (recommended mode) with device independent RGB, indexed by pixel, with 256 colors. Palettes are saved using both the stack method (push/pop) and the store method (by ID) for easy retrieval for later use. Saving palettes to ID numbers (store) is the recommended method and is more efficient than saving palettes to the stack.

This program also shows a common potential error in PCL programs. For color raster images, Foreground color should always be set to black. Otherwise, the Foreground color will mix with all the colors in the raster images, resulting in incorrect color output. Normally index 0 of the palette used with the PCL Imaging mode (RGB) is black, but in this case, index 9 is black. In this case, accidentally setting the Foreground to 0 corrupts the raster image colors resulting in incorrect color output.

Output: This program produces a 1/2-inch by 1-inch rectangle mage up of multiple color vertical lines.

```
EscE    // Reset
Esc*p0P    // Save the current palette to the stack
Esc*v1S    // Set Foreground color to black
This produces a simple color barcode.
Esc&u600D"    // Unit of Measure
Esc*p600X"    // Move CAP Horizontal (PCL Units)
Esc*p600Y"    // Move CAP Vertical (PCL Units)
Esc*v6W    // Configure Image Data (CID)
hex_raw* [ 00 01 08 00 00 00 ]    //select PCL Imaging mode, device
                  // dependent RGB, indexed by pixel, 256 colors
Esc*t600R    // Raster Resolution to 600 dpi
Esc*p0P    // Save the palette to the stack
Esc*r600S    // Set Source Raster Width to 600
Esc*r300T    // Set Source Raster Height to 300
Esc*v0A    // Color Component 1
Esc*v0B    // Color Component 2
Esc*v255C    // Color Component 3
Esc*v0I    // Assign Color Index 0 to (0,0,255)
Esc*v0A    // Color Component 1
Esc*v0B    // Color Component 2
Esc*v0C    // Color Component 3
Esc*v9I    // Assign Color Index 1 to (0,0,0) black
Esc&p4I    // Palette Control ID
Esc&p6C    // Save Palette to ID 4
            // Saving palettes to ID numbers (store) is the recommended
            // method and is more efficient than saving palettes to the
stack

Esc*v9S    // Set Foreground color to black based on the new palette
(index=9)

Esc*r0A    // Start Raster

Esc*b0M    // Set Compression Method to Unencoded

Esc*b600W    // Transfer Raster by Row/Block

hex_raw* [

20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 01 01 01 01
01

01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02 02
02

02 02 02 02 02 02 02 02 02 02 03 03 03 03 03 03 03 03 03 03 03 03 03 03
03

03 03 03 03 03 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04
04

05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 06 06 06 06
06

06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 07 07 07 07 07 07 07 07 07
07

07 07 07 07 07 07 07 07 07 07 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20

20 20 20 20 20 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01

02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 03 03 03 03
03

03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 04 04 04 04 04 04 04 04 04
04

04 04 04 04 04 04 04 04 04 04 05 05 05 05 05 05 05 05 05 05 05 05 05 05
05

05 05 05 05 05 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06

07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 20 20 20 20
20

20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 01 01 01 01 01 01 01 01 01
01

01 01 01 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02 02 02 02 02 02 02
02

02 02 02 02 02 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03
03

04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 05 05 05 05
05
05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 06 06 06 06 06 06 06 06 06
06

06 06 06 06 06 06 06 06 06 06 07 07 07 07 07 07 07 07 07 07 07 07 07 07
07

07 07 07 07 07 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20

01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 02 02 02 02
02

02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 03 03 03 03 03 03 03 03 03
03

03 03 03 03 03 03 03 03 03 03 04 04 04 04 04 04 04 04 04 04 04 04 04 04
04

04 04 04 04 04 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
05

]

Esc*b3M    // Set Compression Method to Delta Row Compression

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block

Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
```

```
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*b0W    // Transfer Raster by Row/Block
Esc*rC    // End Raster
EscE    // Reset
```

Printing a Checkerboard with a Four Color Palette

File Name: CheckrBd

This sample program processes a simple color raster image using the PCL Imaging mode with device independent RGB, indexed by pixel, with a four color palette (2 bits/index). Based on a four color palette indexed by pixel, 00 equals index color 0, 01 equals index color 1, 10 equals index color 2, and 11 equals index color 3. The raster image is then created by hand with each hexadecimal 00 (0000 0000) representing color index 0, 55 (0101 0101) representing color index 1, aa (1010 1010) representing color index 2, and ff (1111 1111) representing color index 3.

Since there is no black in the four color palette, foreground color must be set prior to switching to the four color palette.

Output: This program produces a 1/2-inch by 1/2-inch rectangle mage up of eight by eight matrix of colored boxes.

Note

This program will not work with the HP Color LaserJet 4500 printer. The HP Color LaserJet 4500 printer does not support less than 8 bits per index (Indexed by Pixel) with the CID command.

```
EscE    // Reset
Esc*v1S    // Foreground color
This prints an 8 by 8 checkerboard using a four color palette.
Esc*t600R    // Raster Resolution
Esc&u600D    // Unit of Measure
Esc*p600X    // Move CAP Horizontal (PCL Units)
Esc*p600Y    // Move CAP Vertical (PCL Units)
Esc*t300R    // Raster Resolution
Esc*r128T    // Source Raster Height
Esc*r128S    // Source Raster Width
Esc*p0P    // Push/Pop Palette
Esc*v6W    // Configure Image Data (CID)
hex_raw* [ 00 01 02 00 00 00 ]
Esc*v255A    // Color Component 1
Esc*v255B    // Color Component 2
Esc*v0C    // Color Component 3
Esc*v0I    // Assign Color Index (0 = yellow)
Esc*v0A    // Color Component 1
Esc*v255B    // Color Component 2
Esc*v0C    // Color Component 3
Esc*v1I    // Assign Color Index (1 = green)
```

```
Esc*v0A    // Color Component 1

Esc*v0B    // Color Component 2

Esc*v255C    // Color Component 3

Esc*v2I    // Assign Color Index (2 = blue)

Esc*v255A    // Color Component 1

Esc*v0B    // Color Component 2

Esc*v0C    // Color Component 3

Esc*v3I    // Assign Color Index (3 = red)

Esc*r1A    // Start Raster

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
```

```
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block
```

```
hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]
```

```
Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
```

```
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block
```

```
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff
]
```

```
Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
```

```
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55
aa aa aa aa 00 00 00 00

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block
```

```
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa
]
```

```
Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff
55 55 55 55 aa aa aa aa

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
```

```
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55
]
Esc*b32W    // Transfer Raster by Row/Block
hex_raw* [
aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa 00 00 00 00
ff ff ff ff 55 55 55 55

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block
```

```
hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]
```

```
Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*b32W    // Transfer Raster by Row/Block

hex_raw* [

55 55 55 55 aa aa aa aa 00 00 00 00 ff ff ff ff 55 55 55 55 aa aa aa aa
00 00 00 00 ff ff ff ff

]

Esc*rC    // End Raster Graphics

EscE    // Reset



// Input File Size = 87045 bytes
```

Printing a Checkerboard Raster Pattern

File Name: PgChkSCM

The following example shows how to print a checkerboard raster pattern for each of the four print directions. At the resolution of 75 raster pixels per inch, the size of this raster is 16 by 16. (in PCL units the size is 64 x 64, and in simulator output dots of 600, the size is 128 by 128). This example uses Simple Color mode 3, 3-Plane RGB, with Foreground color set to black.

Output: This program prints text, a simple color raster image, and 3 black rules with differing patterns in each of the four print directions on the page.

```
EscE    // Reset
Esc*r3U    // Simple Color
Esc*v0S    // Foreground color
Esc&a0P    // Print Direction
Esc*p0X    // Move CAP Horizontal (PCL Units)
Esc*p0Y    // Move CAP Vertical (PCL Units)

This test, PageCheckSCM.t, shows that

common ThinPCL objects print correctly
for each of the four print directions.

Print a design using color raster.

The colors are black, red, green, blue.


Esc*r0F    // Raster Presentation

Esc*r64T    // Source Raster Height

Esc*r64S    // Source Raster Width

Esc*r1A    // Start Raster
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*rB    // End Raster Graphics

Print 3 rules; black then patterns 3,4


Esc*c100A    // Horizontal Rectangle Size (PCL Units)
Esc*c150B    // Vertical Rectangle Size (PCL Units)
Esc*c0G    // Pattern ID
Esc*c0P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c3G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c4G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc&a90P    // Print Direction
Esc*p0x    // Move CAP Horizontal (PCL Units)
```

```
Esc*p0Y    // Move CAP Vertical (PCL Units)

This test, PageCheckSCM.t, shows that

common ThinPCL objects print correctly

for each of the four print directions.


Print a design using color raster.

The colors are black, red, green, blue.

Esc*r0F    // Raster Presentation

Esc*r64T    // Source Raster Height

Esc*r64S    // Source Raster Width

Esc*r1A    // Start Raster

Esc*b8V    // Transfer Raster by Plane

hex_raw* [ 00 00 00 00 ff ff ff ff ]

Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
```

```
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*rB    // End Raster Graphics

Print 3 rules; black then patterns 3,4


Esc*c100A    // Horizontal Rectangle Size (PCL Units)

Esc*c150B    // Vertical Rectangle Size (PCL Units)

Esc*c0G    // Pattern ID

Esc*c0P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c3G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c4G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc&a180P    // Print Direction
Esc*p0X    // Move CAP Horizontal (PCL Units)
Esc*p0Y    // Move CAP Vertical (PCL Units)

This test, PageCheckSCM.t, shows that

common ThinPCL objects print correctly
for each of the four print directions.

Print a design using color raster.
The colors are black, red, green, blue.
```

```
Esc*r0F    // Raster Presentation
Esc*r64T    // Source Raster Height
Esc*r64S    // Source Raster Width
Esc*r1A    // Start Raster
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
```

```
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*rB    // End Raster Graphics

Print 3 rules; black then patterns 3,4

Esc*c100A    // Horizontal Rectangle Size (PCL Units)
Esc*c150B    // Vertical Rectangle Size (PCL Units)
Esc*c0G    // Pattern ID
```

```
Esc*c0P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c3G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c4G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc&a270P    // Print Direction
Esc*p0X    // Move CAP Horizontal (PCL Units)
Esc*p 0Y    // Move CAP Vertical (PCL Units)
```

This test, PageCheckSCM.t, shows that
common ThinPCL objects print correctly
for each of the four print directions.

Print a design using color raster.
The colors are black, red, green, blue.

```
Esc*r0F    // Raster Presentation
Esc*r64T    // Source Raster Height
Esc*r64S    // Source Raster Width
Esc*r1A    // Start Raster
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 ff ff ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 00 00 00 00 ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ 00 00 00 00 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ ff ff 00 00 ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff 00 00 ff ff ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
```

```
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff 00 00 ]
Esc*b8V    // Transfer Raster by Plane
hex_raw* [ 00 00 ff ff ff ff ff ff ]
Esc*b8W    // Transfer Raster by Row/Block
hex_raw* [ ff ff ff ff ff ff 00 00 ]
Esc*rB    // End Raster Graphics

Print 3 rules; black then patterns 3,4

Esc*c100A    // Horizontal Rectangle Size (PCL Units)
Esc*c150B    // Vertical Rectangle Size (PCL Units)
Esc*c0G    // Pattern ID
Esc*c0P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c3G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
Esc*p+100X    // Move CAP Horizontal (PCL Units)
Esc*p+0Y    // Move CAP Vertical (PCL Units)
Esc*c4G    // Pattern ID
Esc*c3P    // Fill Rectangular Area
EscE    // Reset
```